Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROYECTO FIN DE CARRERA

TÍTULO DEL PFC: Evaluación de algoritmos de propagación de mensajes en redes oportunistas

TITULACIÓN: Ingeniería de Telecomunicaciones (segundo ciclo)

AUTOR: David García Robles

DIRECTOR: Roc Messeguer

FECHA: 3 de Noviembre de 2013

Título: Evaluación de algoritmos de propagación de mensajes en redes oportunistas

Autor: David García Robles

Director: Roc Messeguer Pallarès

Fecha: 3 de Noviembre de 2013

Resumen

Durante los últimos años la creación y el desarrollo de las redes móviles ad-hoc (MANETs) has sido motivada por la aparición de terminales con dimensiones reducidas y altas capacidades de proceso y conectividad. A su vez, la alta movilidad y versatilidad de estos nodos en las redes MANET ha dado paso a lo que hoy se conoce como Redes Oportunistas. En este tipo de redes los enlaces de conectividad extremo a extremo son altamente variables. Los caminos hacia un destino pueden aparecer o desaparecer repentinamente debido a factores como la movilidad, la energía, etc. Por lo tanto, el retardo en las comunicaciones puede incrementarse considerablemente. Las Redes Oportunistas pueden considerarse como un subconjunto de las Redes Tolerantes a Retardos (DTN como indican sus siglas en inglés).

Dado el contexto anterior, los protocolos de encaminamiento tradicionales para comunicaciones extremo a extremo no son adecuados para este tipo de escenarios. La movilidad, la densidad, las frecuentes desconexiones y las limitaciones propias de los nodos son problemas a tener en cuenta. Existen diferentes propuestas de algoritmos de propagación que intentan maximizar la eficiencia de las transferencias extremo a extremo. Estas aproximaciones se basan en el principio de Custodia-Transporte-Reenvío aprovechando la movilidad y la capacidad de proceso de los nodos de la red, esto provoca un consumo de recursos extra en los nodos intermedios (energía, proceso, almacenamiento, etc.). Diferentes estrategias de propagación a nivel de aplicación han sido propuestas; como por ejemplo *Epidemic* o *Spray and Wait*.

En este documento detalla el estudio del consumo de recursos (energía y espacio de almacenamiento) y la eficiencia (retardo y tasa de entrega del mensaje) para diferentes algoritmos de propagación. Para ello se ha utilizado un simulador basado en NS-3 y escenarios sintéticos.

Palabras clave: *Redes Oportunistas, Redes Tolerantes a Retardo, Algoritmos de Propagación, Energía, Custodia-Transporte-Reenvío, NS-3.*

Title: Evaluación de algoritmos de propagación de mensajes en redes oportunistas

Author: David García Robles

Director: Roc Messeguer Pallarès

Date: November, 3th 2013

Overview

Development of Mobile Ad-hoc NETworks (MANETs) has motivated because the introduction of small intelligent devices with highest process and connectivity possibilities. Moreover, nodes have more mobility and versatility that increase the uncertainty of MANET, so Opportunistic Networks becomes a study case. In an Opportunistic Network the end-to-end links are unstable because network contacts are intermittent and link performance is highly variable or extreme, therefore networks latency can be too high. OppNet it can be consider as a sub-class of Delay Tolerant Networks (DTN).

Given the above context, the traditional end-to-end routing protocols are not suitable for this kind of scenarios. Mobility, density, disruptions and limitations behind nodes are issues to consider. Different proposals routing algorithms has been proposed that attempt to maximize efficiency. These approaches are based on the principle of Store-Carry-Forward (SCF) taking advantage of mobility and the ability to process in intermediate nodes, that strategy has consequent consumption of resources (energy, processing, storage, etc.). Different propagation strategies at the application level have been proposed, such as Epidemic and Spray and Wait.

This paper shows the study of resource consumption in intermediate nodes (energy and storage) and efficiency (message delivery Ratio and delay) for different propagation algorithms. In order to archive our goals we use an NS-3 simulator and synthetic scenarios.

Key-words: *Opportunistic Networks, Delay Tolerant Networks, Routing Algorithms, Energy, Store-Carry-Forward, NS-3.*

AGRADECIMIENTOS

Gracias a mi familia por su cariño y apoyo incondicional en este largo camino.

Agradezco a mi tutor Roc por sus labores de mentor así como por la motivación, el reconocimiento y la dirección del trabajo.

A Jani Lakkakorpi de la universidad Aalto en Helsinki; por su implementación y soporte del código base empleado en este trabajo.

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

El objetivo de este documento es recoger toda la información recopilada durante el Proyecto Final de Carrera. Todos los detalles del trabajo realizado y los resultados obtenidos quedan reflejados en esta memoria.

Este proyecto tiene varios objetivos. El primero es evaluar y comprobar el correcto funcionamiento de la implementación del módulo simulador de *Delay Tolerant Networ*ks (subconjunto de las Redes Oportunistas) realizado por la universidad de Aalto en Helsinki. En segundo lugar se va a editar el código de simulación para añadir la computación del consumo de energía, para ello se va a utilizar el módulo existente en NS-3. En tercer lugar se va a utilizar este simulador y las modificaciones realizadas para evaluar dos de los algoritmos de propagación más comúnmente utilizados en ámbito de estudio de las Redes Oportunistas (*Epidemic* y *Spray and Wait*).

Este documento se organiza en capítulos. En el capítulo uno, este mismo, se presenta una pequeña introducción a la temática de estudio del proyecto, los objetivos principales del trabajo y la organización del propio documento. En el segundo capítulo se detalla una introducción más extendida de los principales conceptos e información necesaria para entender el contenido del resto del documento. En el capitulo tres se especifican las herramientas utilizadas; herramientas de simulación y de computación de datos que se han utilizado para obtener los resultados. Así mismo se detallan los cambios e implementaciones realizadas. En el cuarto capítulo se describe la metodología de trabajo y se presentan los escenarios sintéticos que se han generado para obtener los resultados expuestos en el capítulo cinco. El sexto capítulo presenta la discusión de los resultados obtenidos detallando las conclusiones extraídas para las simulaciones realizadas. Por último, en el séptimo capítulo se detallan las conclusiones generales del proyecto, las implicaciones medioambientales y las posibles líneas futuras derivadas de este trabajo.

CAPÍTULO 2. INTRODUCCIÓN A LAS REDES OPORTUNISTAS

2.1 ¿Qué son las Redes Oportunistas?

Como se define en [1] las Redes Oportunistas (*Opportunistic Networks o OppNets*) pueden ser consideradas como un subconjunto de las Redes Tolerantes al Retardo (*Delay-Tolerant Networks*) donde los nodos son intermitentes y por lo tanto puede no existe un camino de comunicación estable entre origen y destino.

Este tipo de redes se caracterizan típicamente por la intermitencia de los nodos, debido a ello tienen un rendimiento extremadamente variable y son volátiles. Esto significa que durante el ciclo de vida de la red es muy probable que el camino entre origen y destino no siempre esté disponible, o lo que es lo mismo, que este camino aparezca, desaparezca o cambie rápidamente de una manera impredecible. Por lo tanto se puede decir que las oportunidades de comunicación (nodos) son intermitentes.

Dada esta variabilidad es necesario que los nodos intermedios de la red dispongan de capacidad de retención y reenvío de los mansajes a propagar. Esta casuística provoca que las Redes Oportunistas tengan un retardo de propagación elevado y una tasa de reenvío de mensajes variable. Por lo tanto, la mayoría de los protocolos diseñados para comunicaciones *end-to-end* no tienen un comportamiento óptimo. Por ejemplo, es muy probable que las confirmaciones a nivel 4 OSI de los paquetes TCP (*Acknowlegments*) no tengan ruta de vuelta una vez el nodo destino haya recibido el mensaje. Por lo tanto los problemas de congestión debido a las retransmisiones pueden ser muy probables.

Una de las soluciones propuestas a esta problemática es aprovechar de manera oportunista la movilidad de los nodos como medio de propagación. De este modo se pretende maximizar el área de propagación útil de una fuente, ver [2]. Es decir, un nodo intermedio móvil recibe el mensaje, lo custodia y lo retransmite oportunistamente (ver figura 2.1); de esta manera se extiende el área de propagación de la fuente inicial mediante la movilidad de los nodos. Esta aproximación parece una buena manera de hacer llegar el mensaje a nodos lejanos de la fuente origen [3] [4].

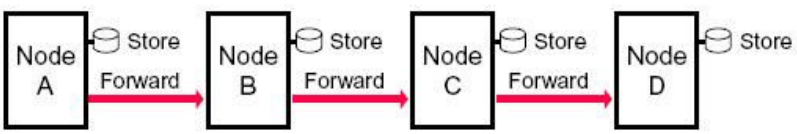

Fig 2.1 – Custodia y envío de mensajes. Fuente [5].

Existen varias soluciones para implementar estas funcionalidades extra de custodia y retransmisión de mensajes en los nodos intermedios. Una de las más aceptadas es implementar una nueva capa intermedia entre los niveles 4 y 5 de la torre OSI de la ISO. Esta nuevas capa es denominada *Bundle Layer* y ofrece capacidades de actuar como nodo, encaminador (*router*) o puerta de enlace (*gateway*). En la figura 2.2 se detallan esquemáticamente las diferencias sigue el encaminamiento de un paquete con routing tradicional (parte de arriba *Internet Transfer*) respecto a el routing (o propagación) utilizado en redes DTN (*DTN Transfers*).

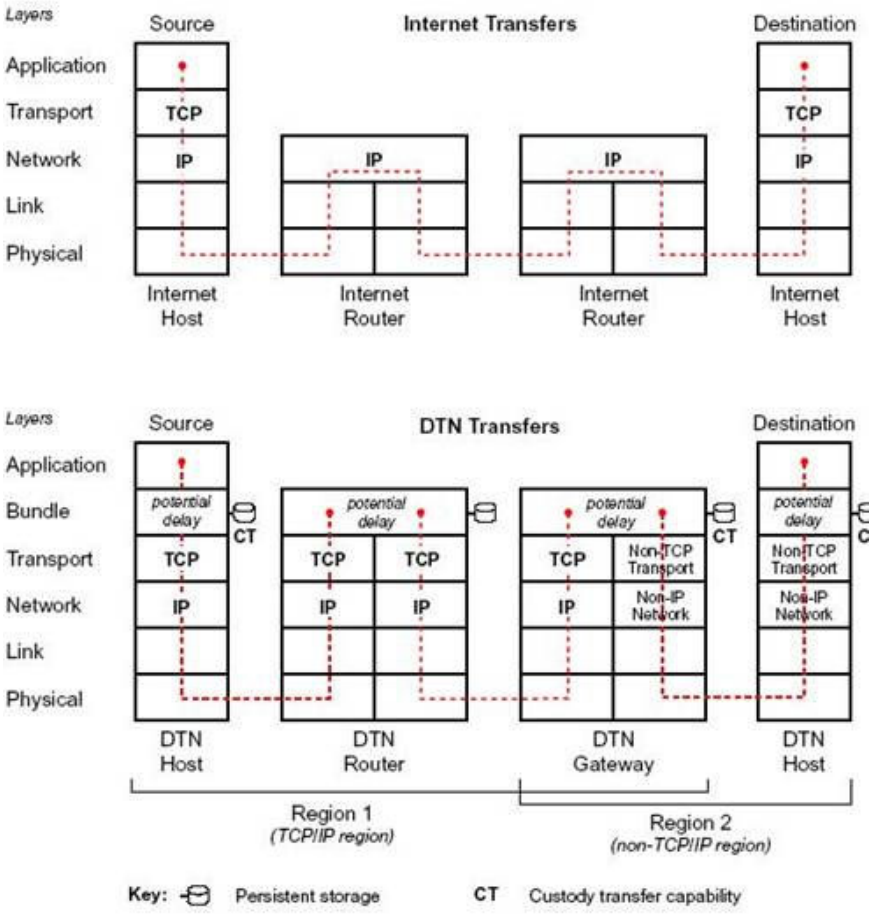

Fig. 2.2 – *Bundle Layer Protocol*. Fuente [5].

Como se puede observar en la figura 2.2 en las redes DTN la decisión de encaminamiento del paquete no se toma a nivel 3 como en el routing tradicional. El paquete sube a un nivel superior (*Bundle Layer*) donde se implementan las funcionalidades específicas de custodia y reenvío. Será esta capa pues quien decida qué hacer con cada uno de los paquetes DTN recibidos en cada nodo.

2.2 ¿Qué son las redes DTN?

La característica principal de las Redes Tolerantes a Retardos, o bien DTN (*Delay Tolerant Network*) como indican sus siglas en ingles; es su elevado retardo. A diferencia de las Redes Oportunistas el factor principal y diferenciador de las redes DTN es el retardo asociado a la comunicación entre origen y destino. Debido a las diferentes problemáticas asociadas a cada uno de los conjuntos y sus aplicaciones; los retos de investigación son tratados de manera independiente.

Por ejemplo en la figura 2.3 se puede observar un ejemplo de red DTN extremo. En enviar un mensaje entre un nodo de la superficie terrestre y un nodo situado en otro planeta de la Vía Láctea pueda suponer un retardo importante. Cabe notar que esta comunicación se puede establecer oportunísimamente o no. Mediante por ejemplo un cohete espacial o aprovechando los movimientos de traslación de los propios planetas.

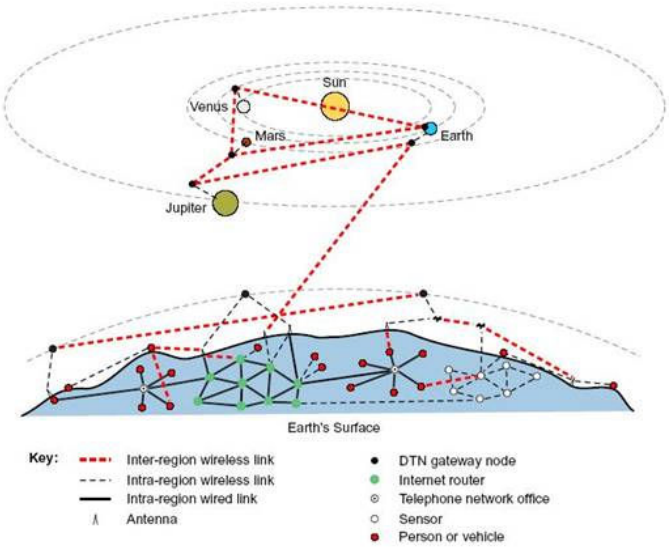

Fig. 2.3 – Ejemplo de Red DTN. Fuente [5].

2.3 Aplicaciones prácticas

Clásicamente las Redes Oportunistas se asocian a entornos y aplicaciones tolerantes a retardos, tasas de error y alta movilidad de los nodos.

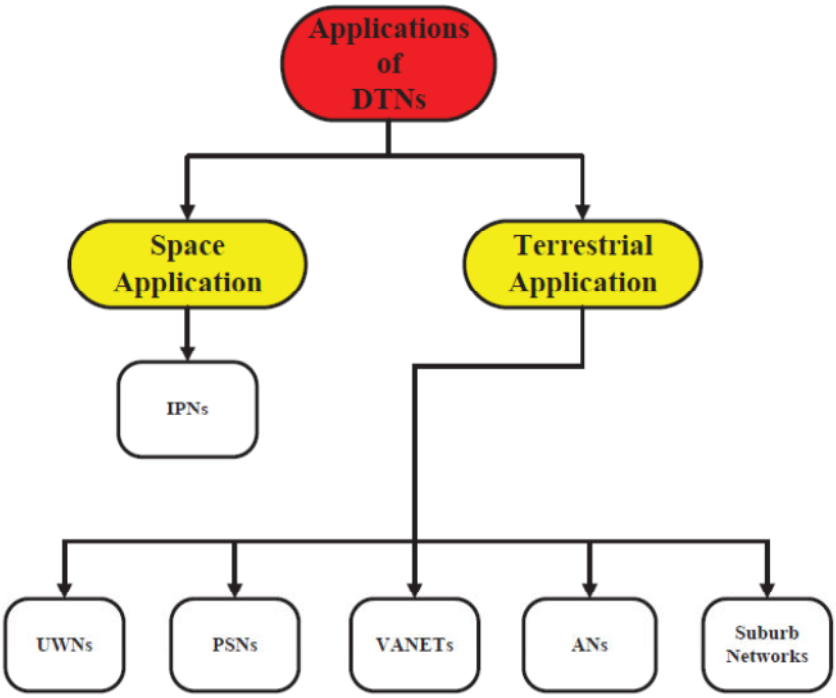

Fig. 2.4 – Aplicaciones de las redes DTN [6].

Como se observa en la figura 2.4 las redes DTN, consecuentemente las Redes Oportunistas, pueden tener varios conjuntos de aplicaciones terrestres:

- *UnderWater Networks* (UWNs) [7]
- *Pocket Switched Networks* (PSNs). [8]
- *Vehicular Ad hoc NETworks VANETs* [9]
- *Airborne Networks (ANs)* [10]
- *Suburb Networks* [11]

Desde un punto de vista más práctico se pueden encontrar ejemplos de aplicación de las Redes Oportunistas en ámbitos militares o en entornos civiles; tales como:

- Difusión de publicidad en entornos comerciales.
- Envío de mensajes en territorio de batalla.
- Seguimiento de animales, plagas, etc.
- Aplicaciones de monitorización del entorno (calidad del aire, climatología, etc.)
- Poblaciones nómadas o *Suburb Networks* (Por ejemplo *Saami Network Connectivity Project* [12].

2.4 Retos de investigación y estudio

Como se ha comentado anteriormente en las Redes Oportunistas es frecuente que los nodos se desconecten para ahorrar batería o se muevan fuera del entorno de cobertura. Esto puede provocar que los enlaces se pierdan y por lo tanto puede hacer que la fuente no pueda alcanzar su desino y el red quede particionada en varias regiones. Para mantener la comunicación entre regiones será necesario que algún nodo intermedio que habilite de manera oportunista la comunicación entre las diferentes regiones de la red. Es por tanto un ámbito de interés en la investigación resolver esta problemática de la manera más eficiente posible. Existen soluciones donde se propone el uso de un nodo con capacidades de transporte (movilidad dentro del escenario) maximizadas (nodos *Data Mule*) [13].

En sí, el principal problema de las OppNets es la capacidad de propagación, muchos estudios relacionados con algoritmos y propiedades de propagación han sido presentados. En [14] se diferencian dos aproximaciones para aprovechar la capacidad de propagación de una red:

- **Por Contacto:** Debido a la movilidad de los nodos y al dinamismo de las redes inalámbricas, los nodos pueden conectarse entre sí de manera impredecible durante un intervalo de tiempo no definido. En entornos donde se pueden predecir la adyacencia entre los nodos es muy recomendable aprovechar esta predicción para habilitar comunicaciones. Routing para Redes Oportunistas [13].
    - *Context-Obiously*
    - *Mobility-Based*
    - *Context-Aware*

- **Por Restricción de Almacenamiento:** Es necesario que los nodos intermedios con funciones de retransmisión dispongan de suficiente capacidad de almacenamiento para evitar pérdidas de información. Estos nodos deben conservar el mensaje durante un periodo indefinido de tiempo, hasta que se alcance el destino u otro nodo tome el relevo. Esta capacidad está claramente condicionada por el número de mensajes que la red debe mantener.

Una vez se han estudiado la capacidad de la red para propagar, es interesante estudiar cómo aprovechar estas capacidades para diseñar un algoritmo que permita alcanzar el destino deseado de la menara más eficiente posible. Una clasificación interesante y generalista de los algoritmos de propagación se propone en [15].

- ***Forwarding-based protocols***: Los nodos nunca mantienen una copia del mensaje, se dedican a retransmitir el mensaje al siguiente salto hasta alcanzar el destino. Es la situación ideal para redes con baja movilidad y retardo o para entornos predecibles, es decir no encaja con la descripción de una OppNet. Generalmente tienen un consumo de recursos bajo.

- ***Replication-based protocols***: Los nodos intermedios mantienen una copia del mensaje y lo replican. Se propagan copias del mensaje original por la red hasta alcanzar el destino. Este tipo de algoritmos se adaptan mejor a entornos muy variables y/o con regiones. Tienen un alto consumo de recursos de la red (memoria de los nodos, capacidad de proceso, batería, ancho de banda etc.), esto provoca un compromiso respecto en la eficiencia del algoritmo en términos de escalabilidad, congestión y uso de recursos.

A continuación se muestra una tabla comparativa de algunos de los algoritmos *replication-based* más utilizados:

Tabla 2.1. Comparativa de *Replication-based protocols*

Algoritmo	Naturaleza	Copias del mensaje en la red	Información contexto	Consumo de recursos	Retardo en la entrega	Prob. entrega
Epidemic	*Flooding-based*	Ilimitado	No	Alto	Bajo	Alta
PRoPHET[1]	*Flooding-based (Data Mule)*	Indeterminado	Si. Las Mulas llevan información sobre la probabilidad de entrega	Medio	Medio	Alta
MaxProp	*Flooding-based* (colas de envio del mnsaje)	Indeterminado	Si. Cada nodo guarda su propio vector de probabilidad de encuentro	Alto	Medio	Alta
RAPID[2]	*Flooding-based* (priorización por paquete)	Indeterminado	Si. Cada paquete contiene su prioridad de envío	Alto	Bajo	Alta
Spray and Wait	*Limited Flooding-based*	Limitado	No	Medio	Medio	Media

Para comparar las múltiples propuestas de algoritmos de propagación existentes se utilizan métricas de medida tales como: la tasa de entrega, el retardo medio de envío, el consumo de energía, el consumo de proceso, el ancho de banda, etc.

El consumo de recursos en la red es un aspecto muy importante a tener en cuenta. Uno de los objetivos de este proyecto es simular y cuantificar el

[1] Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET)
[2] Resource Allocation Protocol for Intentional DTN

consumo de energía que implica utilizar un algoritmo del tipo *Replication-based* en un escenario tipo *Unicast*.

2.5 Algoritmos de propagación utilizados

Los algoritmos de propagación definen la estrategia de custodia y reenvío del mensaje con la finalidad de que éste llegue a su destino. Dependiendo del algoritmo utilizado; la tasa de recepción, el retardo o el consumo de los recursos de la red (energía y capacidad de almacenamiento) pueden verse afectados. Por lo tanto existe un claro compromiso entre el consumo de recursos y la efectividad del algoritmo de propagación.

Se considera mejor algoritmo de propagación aquella estrategia de propagación de datos que consume el mínimo de recursos de la red asegurando alcanzar el destino lo más rápido posible (siempre que exista un camino potencial entre origen y destino).

En [6] se propone una clasificación detallada de los algoritmos de propagación existentes para redes DTN o Redes Oportunistas.

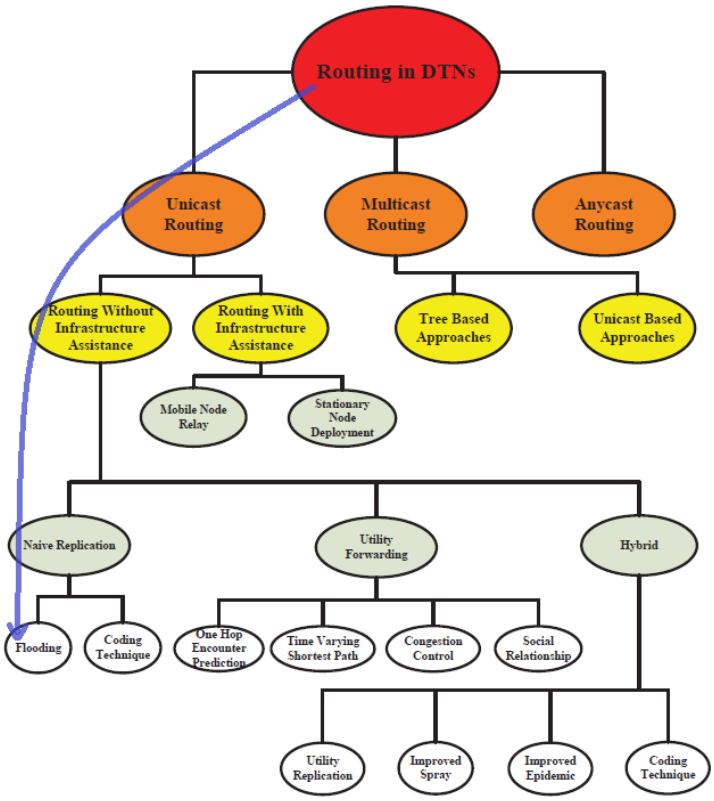

Fig. 2.5 – Clasificación de algoritmos de propagación en Redes Oportunistas. Fuente [6]

La aproximación utilizada en ese proyecto es la propagación por replicación del tipo *Flooding*. En este tipo de propagación se retransmiten copias del mensaje hasta que este llega a su destino (*Replication-based*). Según la clasificación propuesta en la figura anterior estaríamos en el camino: *Routing in DTN – Unicast Routing – Routing without Infrastructure Assistance – Naive Replication – Flooding* tal y como señala la línea pintada en color azul.

A continuación se detallan los dos algoritmos de propagación utilizados en este trabajo; ambos posen la característica *contextless* (sin información de contexto) como se ha podido observar en la tabla 2.1.

2.5.1 Epidemic

El tipo de propagación epidémica se basa en la inundación de la red. Es decir, los nodos retransmiten de manera continuada una copia del mensaje a todos los nuevos vecinos que no dispongan de una. En el escenario más simple; un algoritmo tipo *Epidemic*, no es nada más que una inundación o *flooding* tradicional de la red.

Esta parece una primera solución burda para el problema de propagación que presentan las Redes Oportunistas, es obvio que una mejor aproximación puede llevarse a cabo. Esta aproximación se asegura la máxima efectividad pero por el contrario el consumo de recursos es elevado. Comúnmente, por su simplicidad y elevada tasa de entrega, se utiliza esta estrategia de propagación como referencia a la hora de evaluar un nuevo algoritmo.

Existen aproximaciones más sofisticadas que intentan minimizar el número de copias del mensaje (consumo de recursos de la red) manteniendo una alta tasa de entrega.

2.5.2 Spray and Wait

Esta es aproximación de propagación más sofisticada que la anterior. A diferencia del caso *Epidemic* en *Spray and Wait* se limitan el número de copias del mensaje custodiadas en la red. Fue desarrollado por *University of Southern California* y presentado en el 2005 ACM SIGCOMM.

El principio de funcionamiento se basa en dos fases: Transmisión (*Spray*) y Espera (*Wait*). Cuando un mensaje nuevo se genera en la red, el nodo origen entrega *N* copias del mensaje a sus distintos vecinos (donde *N* limita el máximo número de copias admisibles en la red). Los nodos receptores custodian el mensaje hasta que el destino final sea alcanzable por alguno de ellos.

Se distinguen dos versiones de funcionamiento de este protocolo: *Vanilla* y *Binary*. Las dos versiones se diferencian en como las *N* copias del mensaje se propagan durante el estado de Transmisión (*Spray*). En el modo *Vanilla* solamente se entrega una copia del mensaje a los primeros *N-1* nodos distintos que se encuentra una vez creado el nuevo mensaje. En la segunda versión

(*Binary*), el nodo origen envía *floor(N/2)* copias del mensaje a los vecinos que encuentra. Los nodos receptores custodian las copias y posteriormente, en sus fases de Transmisión, enviarán la mitad de las copias recibidas a cada uno de los vecinos que encuentren siempre que estos no tengan ninguna copia del mensaje. Cuando un nodo elimina propaga todas sus copias (N=1) conmuta a estado *Wait,* donde espera una oportunidad de comunicación directa con el destino.

Tal y como afirman los autores en [16] el principal beneficio de esta estrategia de propagación binaria es que el retardo esperado es optimo para escenarios con nodos que siguen movimientos del tipo IID (*Independent and Identically-Distributed random variables*).

El algoritmo utilizado en este trabajo es la variante *Binary Spray and Wait*.

CAPÍTULO 3. HERRAMIENTAS DE SIMULACIÓN

3.1 Simulador NS-3

Para verificar cualquier teoría sobre el funcionamiento de las redes, o para simplemente probar el funcionamiento de una red particular bajo ciertas condiciones, es conveniente el uso de un simulador. En este capítulo se presentar el simulador utilizado y sus características generales.

Los simuladores de la familia *Network Simulator* (NS-1, NS-2 y NS-3) son una serie de simuladores para redes basados en eventos discretos. El desarrollo de NS empezó en 1989 como una variante del REAL Simulator[3] y actualmente es mantenido por voluntarios sin ánimo de lucro. En este momento NS-2 es la versión más utilizada por la comunidad académica en el área de redes, cuenta por tanto con muchos años de trabajo. Existen una gran variedad de módulos que implementan funcionalidades de red avanzadas. NS-3; web oficial [17] es la última versión, relativamente nueva, de la saga *Network Simulator.* Su primera versión fue creada en 2008. Actualmente le faltan módulos y funcionalidades para equipararse con NS-2, pero se piensa que en un futuro lo suplantará completamente, tal y como NS-2 hizo con NS-1.

3.1.1 Características

El NS-3 es un simulador de redes basado en eventos discretos, programado en su totalidad en C++. Esta dirigido principalmente a la enseñanza y la investigación (NS-3Project, 2011). Es software libre, licenciado bajo la licencia GNU GPLv2. Fue creado y es mantenido por una comunidad voluntaria de investigadores en el área de las redes de comunicación. Su primera versión salió en julio de 2008 y la ultima hasta la fecha (NS-3.18) en agosto de 2013. Si bien es el sucesor del conocido simulador NS-2, el simulador fue creado desde cero, buscando mayor fidelidad con la realidad y mayor orden de las librerías que permitan poder modificarlo con facilidad.

El núcleo de este simulador, al igual que su antecesor el NS-2, está organizado en librerías C++. Cada una de éstas contiene las clases necesarias para simular un programa, protocolo u objeto en particular, como por ejemplo: una aplicación que genera paquetes, una red wifi o un enlace punto a punto. Para simular una red, es necesario crear un script en C++ o en Python en el cual se definen todos los objetos a utilizar, sus parámetros, el vínculo entre ellos y las características generales de simulación.

La implementación utilizada del módulo para Redes Tolerantes a Retardos (*DTN – Delay Tolerant Networks*) de Aalto está probada para la versión 3.16 de NS-3. Esta es la versión que se ha utilizado incluyendo las modificaciones

[3] The REAL Network Simulator - http://www.cs.cornell.edu/skeshav/real/

necesarias en el código DTN para obtener los parámetros relativos a la energía; objeto directo de estudio de este proyecto.

3.1.2 Conceptos básicos: instalación y validación

NS-3 puede correr en diversas plataformas. Para la ejecución de las simulaciones realizadas en este estudio se ha escogido un Sistema Operativo basado en GNU/Linux, Ubuntu [18] en su versión 12.04 LTS para 32 bits.

La instalación de NS-3 es relativamente sencilla para un usuario experimentado con sistemas GNU/Linux. Las notas de instalación se pueden ver en su página oficial [17].

Siguiendo las instrucciones de instalación se puede verificar su correcto funcionamiento lanzando el script *Python* que existe para este uso (*test.py*).

```
./test.py
```

El código de simulaciones programadas con NS-3 típicamente sigue un orden lógico como se representa en el *flowchart* de la figura 3.1.

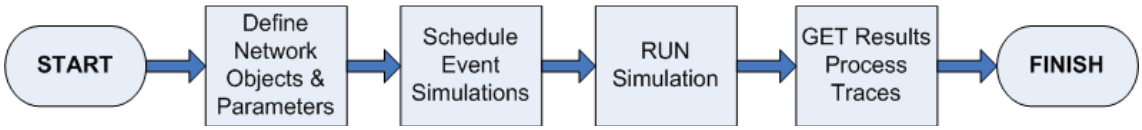

Fig. 3.1 – *Flowchart* para simulación NS-3.

3.2 Implementación DTN de Aalto para NS-3

3.2.1 Aalto

Universidad finlandesa fundada el 1 de enero del 2010; nacida de la fusión de la Universidad Politécnica de Helsinki, la Universidad de Economía de Helsinki y la Universidad de Arte y Diseño Helsinki. Ver [19].

En la implementación y diseño de este código ha sido creado por Jani Lakkakorpi y publicado en el año 2012, está disponible en [20] y es compatible con la versión 3.16 de NS-3.

Actualmente existe una versión de DTN compatible con la versión 3.17 de NS-3, simplemente es una adaptación, no implementa ningún cambio ni modificación respecto la versión anterior.

3.2.2 Conceptos básicos

El simulador DTN de Aalto implementa varias funcionalidades para simular redes DTN y Redes Oportunistas en función de los escenarios y parámetros de entrada con los que se ejecute.

La base del simulador son los mensajes tipo "*Hello"* que se utilizan para mantener una tabla de vecinos y la información de los mismos, como por ejemplo el estado de las colas (importante para el uso del control de congestión). Estos mensajes se envían por defecto cada 100ms y permiten a cada nodo mantener actualizada información de los vecinos que tiene en su alcance de cobertura. La implementación está basada en una arquitectura tipo *bundle protocol* [21] para permitir el transporte de datos de aplicación extremo a extremo utilizando TCP o UDP como protocolos de transporte.

El simulador asegura la correcta recepción de todos los paquetes ya que implementa la retransmisión de datagramas UDP. Por lo tanto, con ambos protocolos de transporte (UDP y TCP) la tasa de pérdida de paquetes es cero. Se recomienda el uso de UDP debido al elevado consumo de recursos de simulación que tiene TCP, pueden existir cientos de conexiones simultáneas dependiendo del escenario, el algoritmo de propagación utilizado y el patrón de movilidad.

Los algoritmos de propagación Epidemic y *Binary Spray and Wait* están soportados y probados por el autor. El código implementa una confirmación de recepción del mensaje a nivel de aplicación.

El código esta implementado en los ficheros *dtn.cc, mypacket.cc* y *mypacket.h* que se pueden descargar juntamente con el manual de instalación desde la página web oficial.

3.2.3 Instalación y validación

La instalación del modulo DTN de Aalto para NS3 es compatible y esta testeada para la versión 3.16 de NS-3. Se puede configurar y compilar el modulo fácilmente siguiendo las instrucciones del autor publicadas en [20]. Dependiendo de la versión de compiladores C y C++ utilizados es posible que la compilación del modulo DTN proporcione algún *warning* y no termine con éxito. Para evitar estos problemas y tener una compilación satisfactoria se deberán deshabilitar el tratamiento de las advertencias de compilación tipo *warning*, una vez ubicados los ficheros de código (ver punto B.2 del documento anterior [20]) ejecutar la compilación con los siguientes parámetros:

```
CXXFLAGS="-Wall" ./waf configure
 ./waf -vv
```

Es recomendable volver a compilar NS-3 deshabilitando igual que anteriormente los *warnings*, es decir en el directorio de /ns-allinone-3.16/ ejecutar:

```
CXXFLAGS="-Wall" ./build.py --enable-examples --enable-tests
```

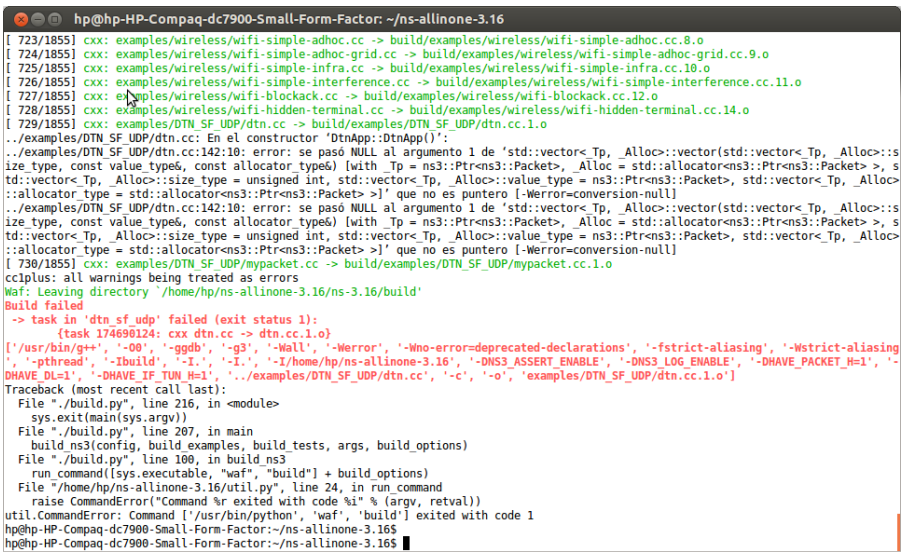

Fig. 3.2 – Error de compilación. *Warning* tratado como *Error*.

Una vez instalado correctamente se pueden ejecutar los scripts de pruebas que aporta el propio autor. En función del hardware utilizado el tiempo de simulación puede alargarse varios días. Para comprobar el correcto funcionamiento del simulador así como entender su estructura se puede revisar el documento [22] y reproducir los resultados obtenidos en el mismo.

3.2.4 Modificaciones realizadas

La implementación base simula todos los aspectos característicos de Redes Oportunistas, poniendo especial hincapié en los algoritmos de propagación basados en la custodia del mensaje y sin información del contexto de la red (*contextless*); ver capítulo 2. Por lo tanto, es totalmente necesario disponer de buffers individuales para cada nodo así como un patrón de movilidad predefinido. El protocolo de transmisión por defecto es UDP. La recepción del mensaje por la fuente destino es confirmada con un mensaje de vuelta (tipo *ACK*) en cada salto; cabe notar que esta confirmación se hace a nivel de aplicación (*Bundle Layer*) y no a nivel IP.

Se ha modificado el código para poder utilizar la opción de envío de *ACK bundle* bajo demanda en cada ejecución; en cada simulación.

El código inicial contempla la computación de paquetes descartados a nivel estadístico, paquetes tipo *antipacket*. Estos paquetes no tienen ningún sentido funcional de cara a la aplicación, son simplemente el almacenamiento de información estadística de cara a un posterior análisis. Este tipo de paquetes simbolizan los paquetes que no se han podido enviar o se han descarado por

restricciones de los parámetros de simulación, por ejemplo la capacidad de los *buffers*. A su vez, este tipo de paquetes *antipacket* se utilizan para el vaciado de memoria de los nodos intermedios. Una vez el nodo destino final recibe el mensaje (*bundle*) envía un paquete tipo *antipacket* con propagación *Epidemic* para que todos los nodos intermedios que vean este paquete dejen de custodiar el mensaje y liberen así su espacio de memoria.

En la implementación base no se ha contemplado ni existe ningún parámetro para controlar el consumo de energía ni recursos de los nodos.

Existe un módulo de energía para NS-3, ver [23]. En este proyecto interesa integrar este módulo en la implementación DTN de Aalto para poder evaluar el efecto de los algoritmos de propagación en el consumo de la energía para Redes Oportunistas. Realizando esta implementación podremos estudiar la energía disponible en cada una de las simulaciones en función de la movilidad, la propagación, etc.

En las simulaciones realizadas, además de computar la energía disponible en el escenario, se ha modificado el código fuente base pare evitar la aparición de *antipackets.* Por lo tanto nos aseguramos que en todas las simulaciones nunca se descartan paquetes por problemas de capacidad en los nodos. Cabe notar que el tiempo de vida del mensaje en la cola no se ha modificado y está fijado a nivel de código a 750 segundos. Al deshabilitar los *antipackets* la funcionalidad de vaciado de colas queda implícitamente deshabilitada.

Las fuentes origen y destino de los *bundles* se han escogido al azar forzando a que el número de saltos para establecer la comunicación sea superior a uno en al menos alguna de las comunicaciones. La fuente origen envía 100 mensajes *bundle* de 1000 bytes cada uno distribuidos aleatoriamente para cada simulación entre [10,(simulation_time-20)] segundos. El tiempo de simulación se ha fijado a 1800 segundos. Pare ver más detalle de la parametrización de las variables del entorno y la simulación consultar el capitulo cuatro, concretamente la tabla 4.2

3.3 BonnMotion: generador de escenarios

BonnMotion es una herramienta programada en Java que permite generar escenarios sintéticos según los diferentes patrones de movilidad y distribución [24]. Está implementada y tiene copyright por la *University of Bonn* en Alemania.

Ésta herramienta es de común uso en la comunidad de investigación para generar escenarios de movilidad diversos. Es compatible con formatos NS-2, que a su vez se pueden cargar en NS-3 utilizando las librerías de compatibilidad. La implementación de DTN de Alto soporta nativamente esta funcionalidad.

Se ha utilizado esta herramienta para generar los escenarios de testeo para validar los cambios realizados en el código fuente del simulador así como para generar los escenarios de simulación probados.

Los escenarios generados se pueden consultar en el capítulo cuarto del presente documento en el apartado 4.2.

3.4 Análisis y computación de los resultados

El simulador está preparado para sacar trazas de la simulación a un fichero de salida tipo *log* donde se registran línea a línea todas las acciones realizadas en el tiempo de simulación. Se ha ampliado el contenido de este registro para poder extraer información referente a la energía del escenario y comprobar que los parámetros modificados (*antipackets* y confirmación de recepción) sean efectivos.

Por tanto, como resultado de cada simulación tenemos un fichero de texto plano tipo *logging* con el registro de eventos y su correspondiente referencia temporal. Este fichero de texto plano puede tener una extensión elevada en función del número de nodos, el tiempo de simulación etc. pudiendo llegar a varios megabytes en simulaciones grandes.

Para extraer los resultados (gráficas y tablas) obtenidos en el capitulo quinto de este trabajo se ha dedicado un esfuerzo elevado a tratar este fichero resultante. La obtención de la información no es directa del simulador y se debe tratar este fichero resultado para extraer la información deseada. Para realizar esta laboriosa tarea se han utilizado herramientas de procesado de datos nativas de Linux, tales como:

- *awk*: filtrado y orden de los resultados.
- *paste*: copia de los resultados en nuevos documentos para analizar.
- *wc*: computo de caracteres
- *sort*: orden de los resultados
- *split*: dividir ficheros
- *gnuplot*: tratado gráfico de datos. Versión 4.4.

CAPÍTULO 4. PLAN DE PRUEBAS

4.1 Metodología del trabajo

Como se ha comentado en el apartado anterior; los escenarios han sido generados con la herramienta BonnMotion. Para el caso de escenarios móviles se ha elegido la movilidad tipo *Random Walk*. Este tipo de movilidad sintética es utilizada en muchos ámbitos de investigación tales como la medicina, la biología, el procesamiento de imágenes, etc. Es por tanto un patrón de movilidad extendido en el ámbito de la investigación. Básicamente consiste en calcular la posición futura de un nodo dependiendo sólo de su posición en el instante previo y alguna variable aleatoria determinando así su subsecuente dirección y la longitud de avance.

Para la evaluación de los cambios implementados en este proyecto sobre el código de DTN se han utilizado escenarios estáticos. Estos escenarios son más predecibles y permiten analizar los resultados más fácilmente. Para validar las modificaciones realizadas han sido necesarias muchas simulaciones. Implícitamente a los resultados de este proyecto existe por tanto una importante inversión de tiempo tanto en simular, interpretar y analizar los datos obtenidos con la finalidad de corregir, depurar y verificar los cambios realizados en el código. Esta técnica de ensayo y error ha servido para validar el correcto funcionamiento de los cambios realizados. A su vez ha permitido coger experiencia con las simulaciones, y lo que es más importante, obtener unos primeros resultados exploratorios para observar el comportamiento del simulador. A partir de aquí se han ido seleccionando los parámetros de evaluación que son de interés para este trabajo (ver apartado 4.1.2).

Las simulaciones estáticas han servido como pruebas exploratorias para obtener los primeros resultados base, que a su vez permiten tener más información a la hora de decidir que entornos de simulación son interesantes en un escenario más complejo, con movilidad. Los resultados obtenidos con escenarios móviles requieren un mayor esfuerzo de análisis y a su vez aportan posibles líneas futuras de investigación derivadas de este trabajo.

4.1.1 Hardware y software complementario

En la tabla 4.1 se detallan los parámetros del software complementario a la instalación del simulador NS-3 y el código de simulación de DTN. Estos parámetros no son relevantes de cara a interpretar los resultados obtenidos pero si de cara a reproducir las pruebas realizadas.

Tabla 4.1 – Parámetros de soporte a la simulación

Parámetro	Valor
Hardware	
Procesador	Intel(R) Core(TM)2 Duo CPU E8500@3.16GHz
Memoria	4GB
Disco Duro	SATA 500GB cache:16MB
Software	
Sistema Operativo	Ubuntu 12.04 (precise) 32 bits
Kernel	Linux 3.2.0-41-generic-pae
Versión NS-3	3.16
Versión DTN	Compatible con NS-3 3.16
Versión compilador	GCC 4.6.3
Versión GNUPlot	4.4 pachlevel 3

En este entorno de trabajo las simulaciones realizadas, las de mayor número de nodos y complejidad, han tardado del orden de 3-4 días. Notar que el valor más importante para reducir el tiempo de simulación es la CPU y por defecto un proceso (una simulación) solo utiliza una de las dos CPU disponibles en el hardware utilizado.

4.1.2 Parámetros a evaluar

En este proyecto se han tomado varios parámetros para analizar el comportamiento de cada algoritmo los escenarios propuestos. A continuación se muestra el listado de parámetros resultantes obtenidos para cada simulación:

- ***Available Energy***: Energía disponible en el escenario. Suma de la energía disponible por cada uno de los nodos que componen el escenario de simulación. Normalmente se muestra su evolución en función del tiempo.

- ***Delivery Delay***: Tiempo que tarda el mensaje en alcanzar su destino final. Retardo total provocado por la propagación, transmisión etc. Los mensajes retransmitidos se penalizan con un retardo extra de 1000 segundos a nivel de simulación. En los resultados mostrados para este parámetro se omiten los mensajes retransmitidos puesto que no se consideran relevantes para el estudio realizado. Normalmente se presenta este parámetro en forma de histograma de probabilidades.

- ***Delivery Ratio***: Tasa de entrega de los mensajes. En el caso ideal debería ser del 100%, es decir todos los mensajes que envía la fuente origen llegan a su destino; no hay pérdidas por incomunicación.

- ***Number of Messages in Environment (Memory usage)***: Número de mensajes totales (incluyendo las copias del mensaje original) que hay en el escenario. Normalmente se presenta su evolución en el tiempo de

simulación. Este parámetro se puede ver como la ocupación de memoria RAM interna que la red debe gastar en almacenamiento para propagar el mensaje. Cabe notar que el simulador DTN tiene un *timeout* de custodia del mensaje por cada nodo fijado a 750 segundos, es decir al cabo de 750 segundos de entrar un mensaje en la cola será eliminado.

4.1.3 Parámetros fijados

NS-3 permite cambiar infinidad de parámetros. En las simulaciones realizadas en este trabajo se han realizado algunos cambios respecto a los parámetros por defecto base del simulador NS-3. Además se han añadido otros parámetros de simulación referentes al contexto implementado (propios del código DTN y de las implementaciones realizadas); como por ejemplo: el tiempo de vida (tiempo de custodia) de los mensajes o la energía inicial de los nodos.

Cabe destacar que el medio físico utilizado es, obviamente, Wifi (802.11g). En este caso no se ha modificado el modelo de propagación ni su tasa de error. De esta manera aseguramos que todos los nodos dentro del radio de cobertura del emisor captarán los mensajes. Los parámetros físicos concernientes al medio de transporte no son de interés para este estudio, su modificación ni aleatoriedad aportan datos objetivos para el análisis. Por lo tanto no se han modificado sus valores por defecto.

En la siguiente tabla se muestran los parámetros modificados respecto a los valores por defecto de NS-3 así como los parámetros propios del simulador DTN de Aalto.

Tabla 4.2 – Valores intrínsecos al entorno de simulación.

Parámetro	Valor	Comentarios
Parámetros del Medio Acceso		
Wifi Channel	802.11g	Estándar 802.11g
TxPowerStart/TxPowerEnd	12.5 dbm	Valor por defecto implementación DTN Aalto. Máximo nivel de Tx/Rx
RxFigureNoise	7	Valor por defecto NS-3
EnergyDetectionThreshold	-74.5 dbm	Valor de detección de señal física por defecto en el simulador DTN
CcaMode1Threshold	-77.5 dbm	Valor de señal física ocupada por defecto en el simulador DTN
RxGain / TxGain	1 dB	Ganancia de la antena en Rx/Tx por defecto en Ns-3
Parámetros del simulador DTN		
Bundle_Queue	100000 packets	Buffer de paquetes para cada nodo. Suficientemente grande para no tener *antipackets*
MAC_Queue - Maxpacket	1000 packets	Suficientemente grande para no tener perdidas
Antipaquets Queue	0 packets	Evitar *antipackets*

Bundle lifetime	750 s	Tiempo de vida de los bundle en cola para cada nodo
Hello messages interval	100 ms	Tiempo de refresco de información de vecinos
Simulation time	1800 s	Tiempo de simulación
Hello message limit	2280 bytes	Limite del mensaje de refresco de vecinos
Transport IP protocol	UDP	Confirmación de bundle en capa aplicación
Retransmission timeout	1 s	Retardo añadido por retransmisión del paquete. Por defecto en DTN
Congestion Control	Allways off	Control del estado de la cola del vecino, apagado en todas las simulaciones
Initial_Energy	100 J	Energía inicial para cada nodo, suficiente para que no se acabe
TxCurrentA	0.0174 A	Consumo de Tx por defecto
RxCurrentA	0.0197 A	Consumo de Rx por defecto
IdleCurrentA	0.000426 A	Consumo batería en estado idle
CcaBusyCurrentA	0.000426 A	Consumo en estado ocupado
SwitchingCurrentA	0.000426 A	Consumo para cambios de estado
Parámetros de los datos *bundle*		
Packet bundles size	1000 bytes	Tamaño fijo de los *bundles* para todas las simulaciones
Number of Tx	100	100 envíos aleatorios para cada simulación

En la lista reflejada en la tabla 4.2 aparecen los parámetros más relevantes; cualquier parámetro que no quede reflejado en la tabla anterior actúa con su valor por defecto para la versión 3.16 de NS-3. Consultar [17] para más información.

4.2 Escenarios

A continuación se presentan los escenarios creados para las pruebas realizadas en el presente documento. Todos los escenarios, tanto estáticos como dinámicos se han limitado una extensión física de 300x300 metros y han sido creados con la herramienta BonnMotion presentada anteriormente. En algún caso específico se han utilizado escenarios más grandes, como por ejemplo para forzar tasas de entrega inferiores al 100% (ver punto 5.2.3).

Los resultados de las pruebas realizadas se muestran en el quinto capítulo.

4.2.1 Escenarios estáticos

Se han generado diferentes escenarios estáticos con la finalidad de hacer pruebas exploratorias de la herramienta de simulación y las modificaciones realizadas en la misma.

A continuación se muestran los escenarios estáticos utilizados:

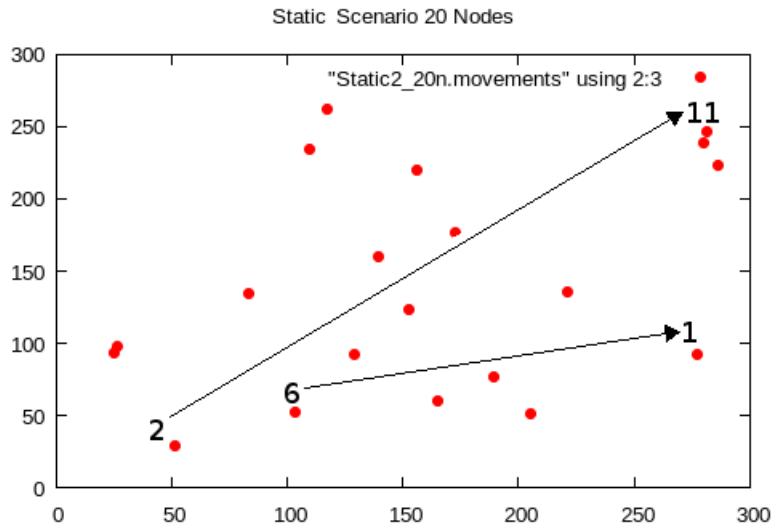

Fig. 4.1 - Escenario con 20 nodos estáticos situados aleatoriamente.

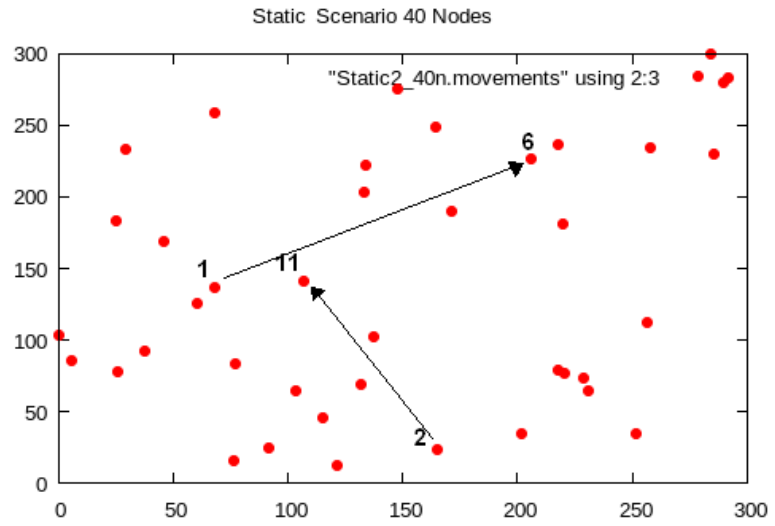

Fig. 4.2 - Escenario con 40 nodos estáticos situados aleatoriamente.

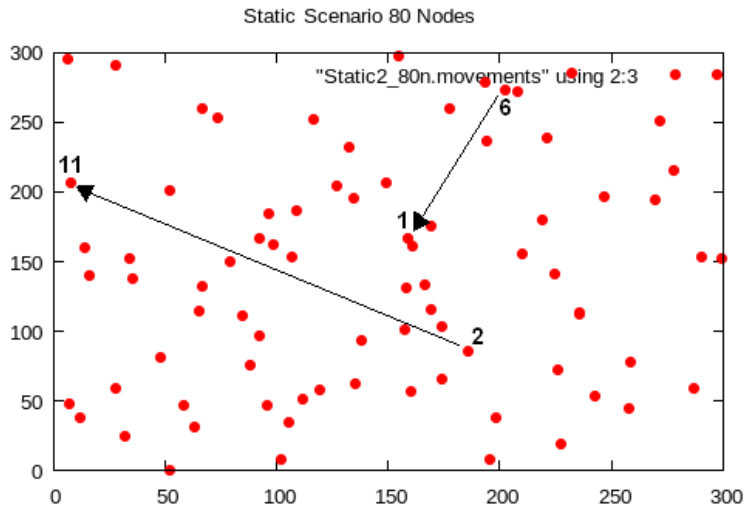

Fig. 4.3 - Escenario con 80 nodos estáticos situados aleatoriamente.

4.2.2 Escenarios con movilidad (*Random Walk*)

Se han generado diferentes escenarios de movilidad para evaluar los parámetros de estudio del presente proyecto. De diferente manera que para los escenarios estáticos; el área de simulación no se ha limitado a 9 hm^2. Se han realizado simulaciones complementarias con un área mayor para verificar parámetros tales como el *Delivery Delay*. El patrón de movilidad utilizado en todos los casos es *Random Walk* con velocidades comprendidas entre 0,5 y 1,5m/s, el tiempo máximo de pausa son 60 segundos. Se ha forzado que cada nodo cambie de rumbo cada 60 segundos.

En la tabla 4.3 se detallan los valores utilizados para el movimiento tipo *Random Walk* utilizado.

Tabla 4.3 – Parámetros de *Random Walk* para escenarios con movilidad.

Parámetro	Escenario	*Data Mule*
Movement Type	*Random Walk*	*Random Walk*
Max. Pause time	60 s	60 s
Nodes Walk until (Mode t)	60 s	60 s
Max. speed	1.5 m/s	4 m/s
Min. speed	0.5 m/s	3 m/s

Un caso particular de movilidad de nodos es el uso de *Data Mule*, donde típicamente se aprovecha la alta velocidad de un nodo (en comparación con el resto de nodos del escenario) para maximizar el efecto de *Store and Fordward* de los datos. De esta manera se espera una mejora en la propagación de los datos por el escenario. Para simular este tipo de escenarios se ha añadido a los escenarios requeridos un nodo con una velocidad que oscila entre 3 y 4m/s,

siendo significativamente superior a la del resto de nodos (entre 0,5 y 1,5m/s). Ver figura 4.7.

En las siguientes figuras se representan gráficamente los escenarios diseñados para las simulaciones con movilidad con 20, 40 y 80 nodos siguiendo un patrón de movimiento tipo *Random Walk*.

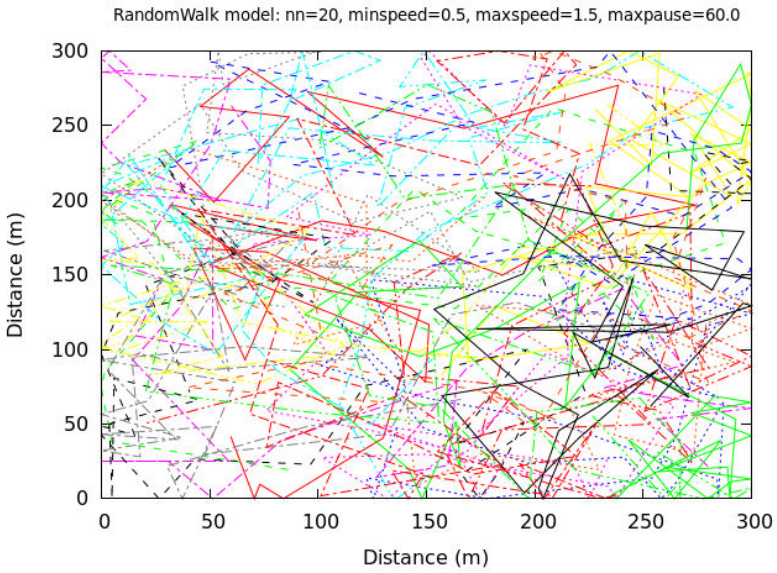

Fig. 4.4 - Escenario con 20 nodos móviles tipo *Random Walk*.

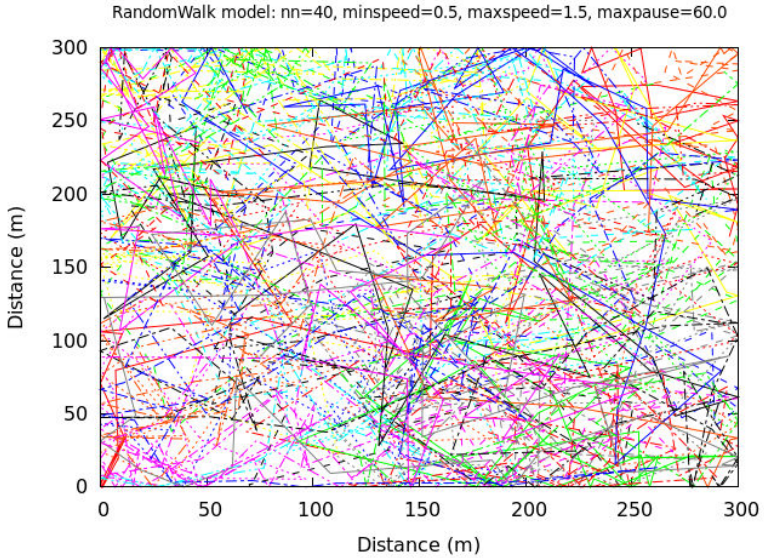

Fig. 4.5 - Escenario con 40 nodos móviles tipo *Random Walk*.

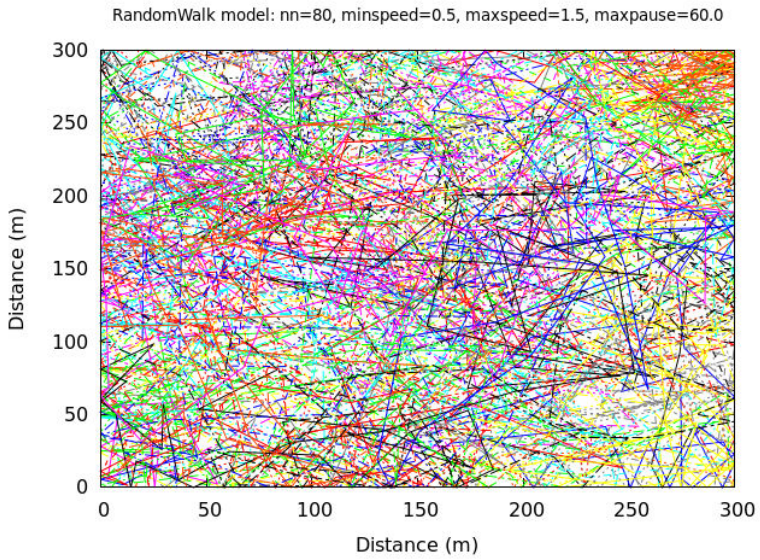

Fig. 4.6 - Escenario con 80 nodos móviles tipo *Random Walk*.

Como se observa en las figuras 4.4, 4.5 y 4.6 la densidad de nodos aumenta considerablemente y esto permitirá evaluar su efecto en los parámetros descritos anteriormente.

La figura 4.5 muestra la representación gráfica del movimiento del nodo *Data Mule* utilizado en el escenario base de 300x300 metros (9 hm^2).

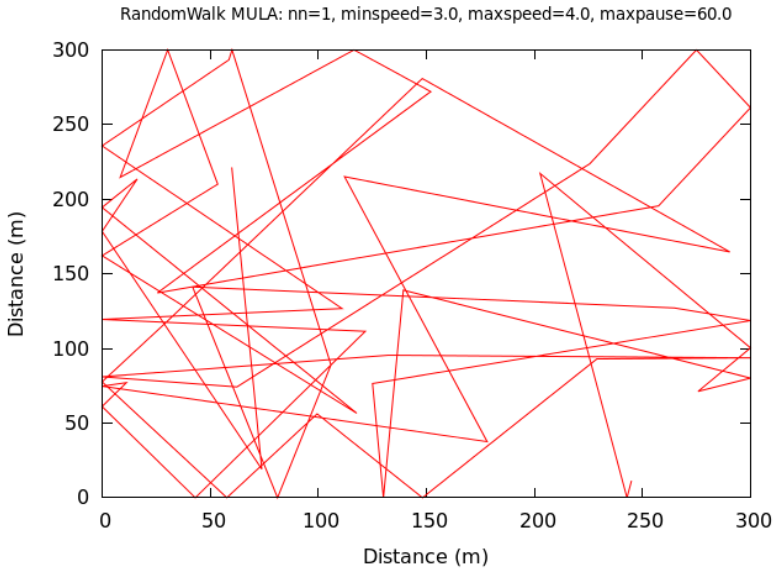

Fig. 4.7 – Representación del nodo *Data Mule*.

Por lo tanto la superposición de los escenarios mostrados en la figura 4.5 y 4.7 será el utilizado en las simulaciones móviles con *Data Mule* para escenarios con área 300x300 metros.

En las simulaciones complementarias realizadas, por ejemplo con escenarios de mayor dimensión, se han utilizado estos mismos valores como referencia. Se ha modificado el tamaño del área de simulación en un entorno de 20 nodos obteniendo los escenarios que se muestran en las figuras 4.8 y 4.9.

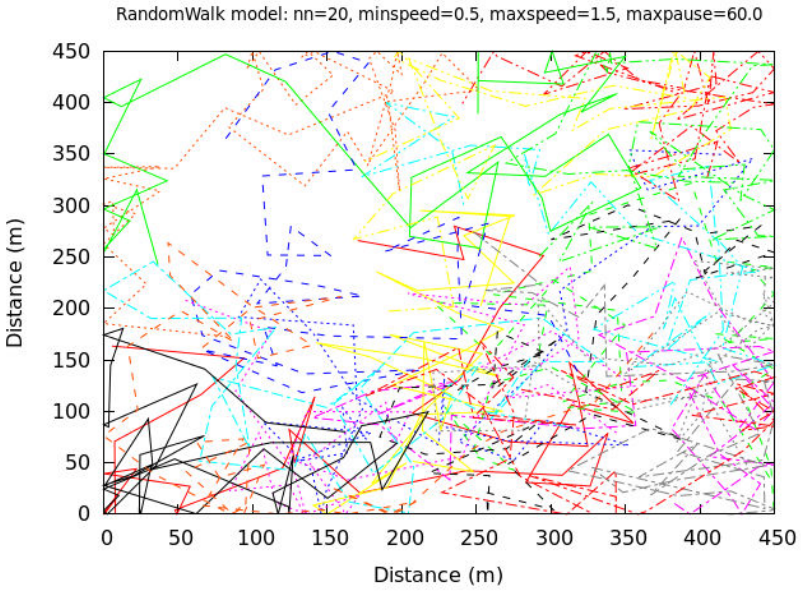

Fig. 4.8 – Escenario de 20,25 hm^2 con 20 nodos móviles tipo *Random Walk*.

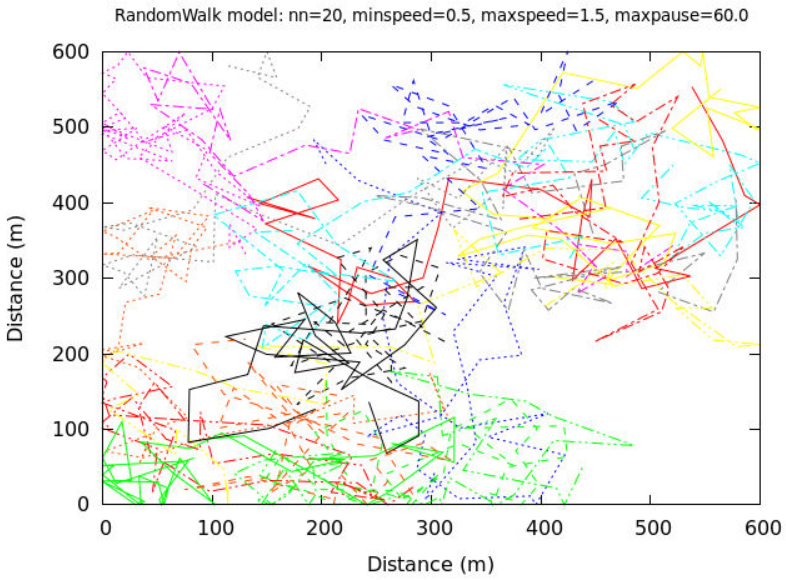

Fig. 4.8 – Escenario de 36 hm^2 con 20 nodos móviles tipo *Random Walk*.

Como se puede ver en las figuras anteriores (4.4, 4.8 y 4.9) la densidad de nodos se decrementa considerablemente favoreciendo así que parámetros como el *Delivery Delay* se vean devaluados. Ver apartado 5.2.3.

Para tratar de mejorar el *Delivery Delay* en escenarios con una densidad de nodos baja y escasas oportunidades de conectividad (ver figura 4.8) se ha introducido un nodo tipo *Data Mule* con la finalidad de mejorar los resultados obtenidos maximizando las posibilidades de comunicación oportunista. En la figura 4.9 se muestra el nodo *Data Mule* utilizado en el escenario de 36 hm^2.

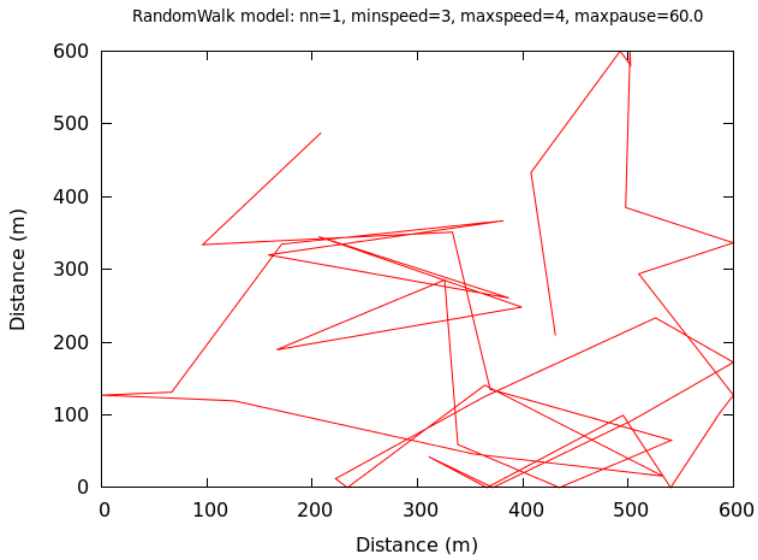

Fig. 4.9 – Representación del nodo *Data Mule* para el escenario de 36 hm^2.

CAPÍTULO 5. PRESENTACIÓN DE RESULTADOS

5.1 Resultados con escenarios estáticos

En este apartado se muestran los resultados obtenidos para los escenarios estáticos descritos en el capítulo 4. En la mayoría de casos, por un problema de espacio, se muestran solamente los resultados más relevantes indicando en que apartado de los anexos se pueden consultar los resultados complementarios.

La descripción de cada uno de los parámetros de análisis se ha descrito anteriormente en el apartado 4.1.2.

Los parámetros de simulación comunes se pueden observar en la tabla 4.1 del cuarto capítulo del presente documento. En la presentación de los resultados solo se indican los parámetros deferenciales de dicha tabla si los hubiera.

Por defecto, si no se indica lo contrario en la información del pie de figura o en la propia imagen, los datos presentados son sin uso de ACK.

La discusión de estas gráficas queda detallada en el capítulo quinto.

5.1.1 Available Energy

En este apartado se presenta la evolución de la energía disponible en el escenario respecto al tiempo de simulación. En la figura 5.1 Se muestra la comparativa para los diferentes algoritmos de propagación probados para el escenario con 80 nodos. El resultado del resto de escenarios descritos en el capítulo 4 se puede consultar en los anexos (ver A.I).

La línea de evolución pintada en color rojo simboliza el consumo de referencia en un entorno sin mensajes de datos. Por lo tanto, esta línea muestra el consumo de energía intrínseco al entorno. Dicho con otras palabras; mantener la vecindad entre los nodos del escenario estático propuesto con 80 nodos durante 1800 segundos consume aproximadamente un 7% de la energía disponible (aproximadamente un 0.0875% de media para cada nodo). Esta energía es consumida por los mensajes de control tipo *Hello* que permiten mantener la tabla de vecindades; mantener la conectividad del escenario viva.

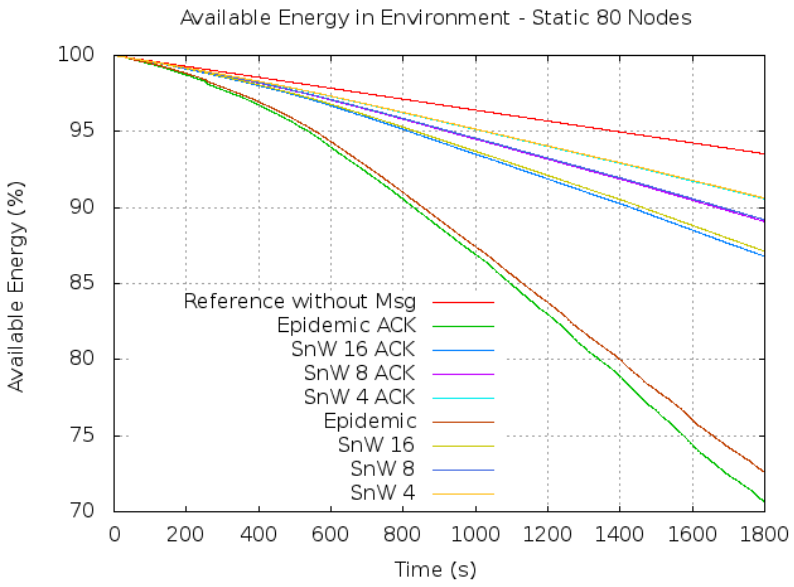

Fig. 5.1 – Evolución de la energía disponible en el entorno.

La tabla 5.1 muestra numéricamente la energía disponible en el escenario una vez finalizados los 1800 segundos de simulación para cada uno del los algoritmos de propagación y los escenarios propuestos en el capítulo 4.

Tabla 5.1 – Datos para la energía disponible en los escenarios estáticos.

Algoritmo de Propagación	**Energía disponible en el entorno (%)**		**Diferencial**
	Sin ACK	**Con ACK**	
80 Nodos			
Epidemic	72.6	70.65	1.95
SnW16	87.12	86.79	0.33
SnW8	89.19	89.08	0.11
SnW4	90.62	90.57	0.05
40 Nodos			
Epidemic	86.52	86.01	0.51
SnW16	90.35	90.07	0.28
SnW8	92.30	92.19	0.11
SnW4	93.82	93.78	0.04
20 Nodos			
Epidemic	90.82	90.56	0.26
SnW16	91.57	91.47	0.10
SnW8	92.94	92.88	0.06
SnW4	94.19	94.16	0.03

5.1.2 Delivery Delay

Como se ha comentado anteriormente, en las figuras tipo *Delivery Delay* se omiten los paquetes retransmitidos. Los mensajes retransmitidos son penalizados a nivel de simulación con 1000 segundos de retardo añadido.

En la figura 5.2 se muestra el histograma para la probabilidad de entrega en cada uno de los algoritmos evaluados en el escenario estático con 80 nodos con y sin uso de ACK. Los histogramas para los escenarios restantes con diferentes densidades de nodos se pueden ver en el anexo A.II.

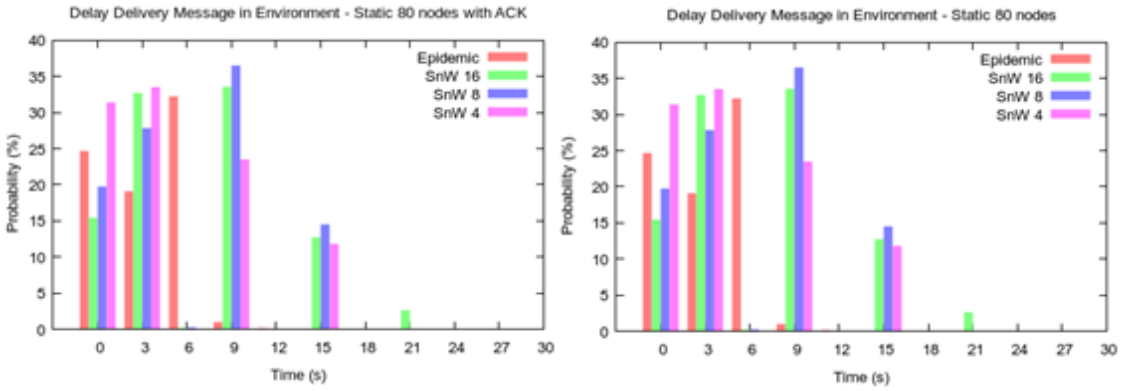

Fig. 5.2 – Retardo de recepción del mensaje.

5.1.3 Delivery Ratio

Otro de los parámetros a tener en cuenta es la tasa de entrega. En este punto se presentan las tasas de entrega para las simulaciones realizadas. En el caso de escenarios estáticos ha sido del 100% para todas las simulaciones realizadas (con y sin ACK). Tal y como se ha comentado en capítulos anteriores se ha asegurado que existe como mínimo un camino viable entre la fuente y el receptor para obtener resultados exploratorios consistentes.

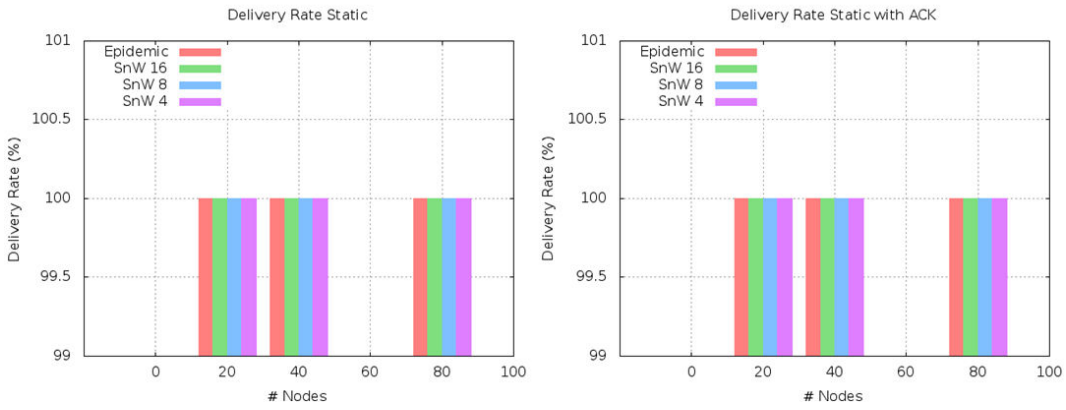

Fig. 5.3 – Evolución de la tasa de entrega.

5.1.4 Number of Messages in Environment

La gráfica de la figura 5.4 permite comprobar el comportamiento de ambos algoritmos respecto a la ocupación de memoria total del entorno. En el caso de *Spray and Wait* se ve claramente que la ocupación se mantiene contenida respecto a *Epidemic* que sigue un aumento constante. Este comportamiento se observa gracias a que la funcionalidad de vaciado de las colas esta deshabilitada (deshabilitar *antipackets* como se explica en el punto 3.2.4).

En la figura 5.4 se muestra el resultado más significativo, 80 nodos. El resto de gráficas se pueden consultar en el anexo A.III.

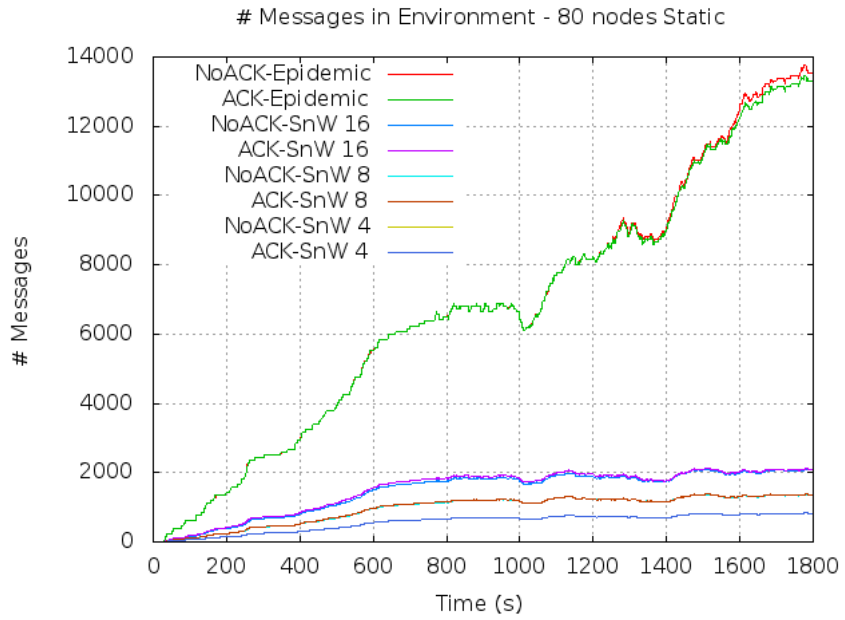

Fig. 5.4 – Evolución de la ocupación de memoria en el entorno.

Para observar mejor las diferencias en cada uno de los escenarios en la tabla 5.2 se muestra el resumen detalle numérico resultante al final de todas las simulaciones realizadas.

Tabla 5.2 – Datos para el número de mensajes en escenario estático.

Algoritmo de propagación	Ocupación de memoria en el entorno (# mensajes)	
	Sin ACK	Con ACK
80 Nodos		
Epidemic	13460	13226
SnW16	2049	2066
SnW8	1330	1342
SnW4	816	816

40 Nodos		
Epidemic	6280	6280
SnW16	1936	1967
SnW8	1221	1228
SnW4	702	704
20 Nodos		
Epidemic	3368	3368
SnW16	1638	1628
SnW8	1188	1192
SnW4	816	814

5.2 Resultados con escenarios móviles

En este apartado se muestran los resultados obtenidos para los escenarios móviles descritos en el cuarto capítulo. Igual que para los resultados estáticos se presentan los resultados más relevantes indicando en que apartado de los anexos se pueden consultar los resultados complementarios.

La descripción de cada uno de los parámetros de análisis se ha descrito anteriormente en el apartado 4.1.2.

Dado el análisis de los resultados exploratorios con escenarios estáticos, expuesto en el apartado 6.1, en el caso de escenarios móviles se ha optado por hacer todas las simulaciones sin ACK con un escenario base de 40 nodos.

Los parámetros de simulación comunes se pueden observar en la tabla 4.1 del cuarto capítulo. La discusión de estos resultados se presenta en el sexto capítulo del presente documento.

Para complementar los resultados obtenidos en escenarios móviles se ha añadido un nodo tipo *Data Mule*. La descripción del nodo *Data Mule* así como los parámetros escogidos para el mismo se pueden observar en el capítulo cuarto (ver apartado 4.2.2 y tabla 4.1).

Los resultados complementarios se muestran en los anexos, queda indicada la referencia exacta en cada uno de los parámetros presentados a continuación.

5.2.1 Available Energy

La figura 5.5 presenta la evolución temporal de la energía disponible en el entorno para los escenarios móviles propuestos. Además, como se ha comentado anteriormente se añade un nodo tipo *Data Mule* para analizar su efecto en los resultados.

De igual manera que para el escenario estático; la línea de evolución de color rojo muestra el consumo de referencia provocado por los mensajes de control del escenario (mantenimiento de las vecindades, etc.). Se considera por tanto que el algoritmo ideal, en lo que a consumo de energía se refiere, sería el más próximo a esta línea.

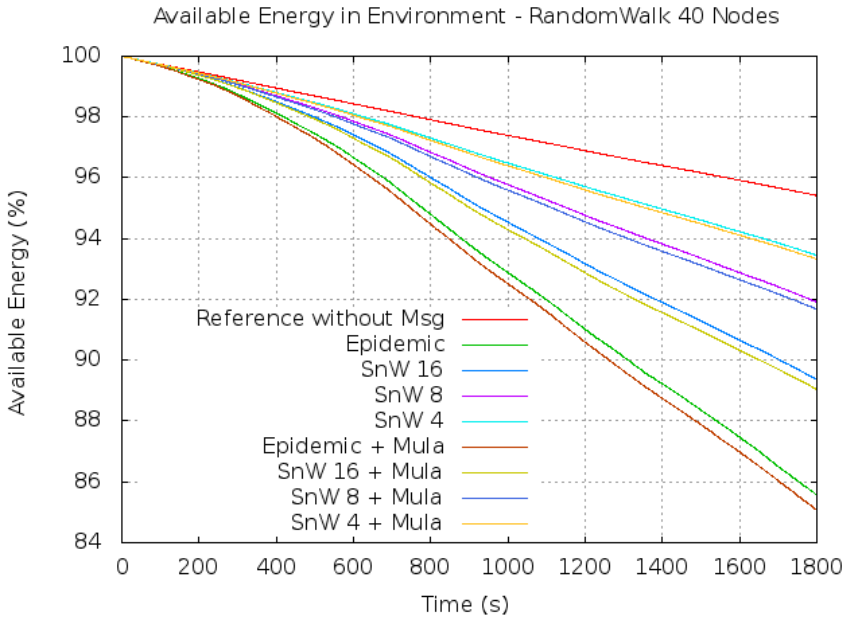

Fig. 5.5 – Evolución de la energía disponible en el entorno móvil.

La tabla 5.3 que se muestra a continuación detalla numéricamente la energía disponible en el escenario una vez finalizados los 1800 segundos de simulación.

Tabla 5.3 – Datos para la energía disponible en escenario móvil.

Algoritmo de Propagación	**Energía disponible en el entorno (%)**		**Diferencial**
	Sin *Data Mule*	**Con *Data Mule***	
Epidemic	85.57	85.08	0.49
SnW16	89.37	89.05	0.32
SnW8	91.91	91.68	0.23
SnW4	93.45	93.33	0.12

Las gráficas para los escenarios con 20 y 80 nodos se pueden ver en el apartado B.I de los anexos.

De manera complementaria a los resultados obtenidos se ha realizado una simulación activando los *antipackets* y las funcionalidades que este tipo de paquetes aportan (vaciado de memoria). En la gráfica que se muestra a continuación se presentan los resultados obtenidos para el escenario con 40 nodos con *antipackets,* sin uso de *Data Mule* y con propagación tipo *Epidemic*.

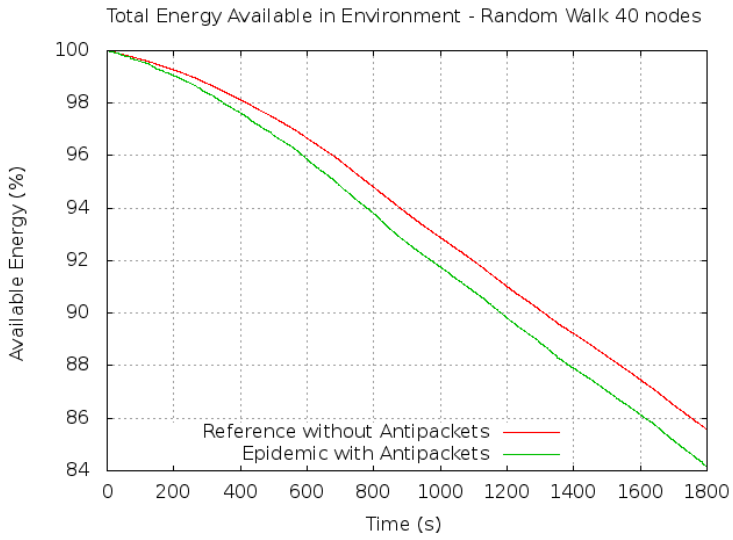

Fig. 5.6 – Evolución de la energía disponible en el entorno móvil con *Antipackets*.

5.2.2 Delivery Delay

En este apartado se muestra el histograma para el retardo de entrega del mensaje en función del algoritmo de propagación.

Igual que para el escenario estático se debe tener presente que los mensajes retransmitidos son penalizados con un retardo extra de 1000 segundos. Estos mensajes quedan fuera del ámbito de representación y de interés de este estudio.

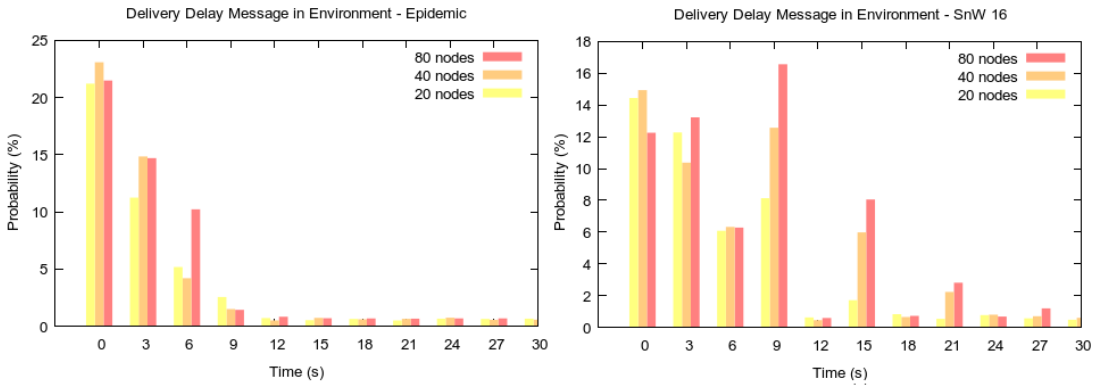

Fig. 5.7 – Histograma del retardo de recepción del mensaje en entorno móvil.

En el apartado B.II se detallan los resultados individuales para los escenarios con 20, 40 y 80 nodos. En el caso de 20 y 80 nodos no se han realizado simulaciones con *Data Mule* y para el caso del escenario de 40 nodos además se han realizado simulaciones con diferentes niveles profundidad de *Binary Spray and Wait, 16, 8,* y *4.*

5.2.3 Delivery Ratio

Sin dunda la tasa de entrega es un parámetro importante para evaluar la eficiencia de un algoritmo. Igual que para el caso de escenarios estáticos; los resultados obtenidos siempre han sido del 100%, es decir todos los mensajes han sido entregados a su destinatario.

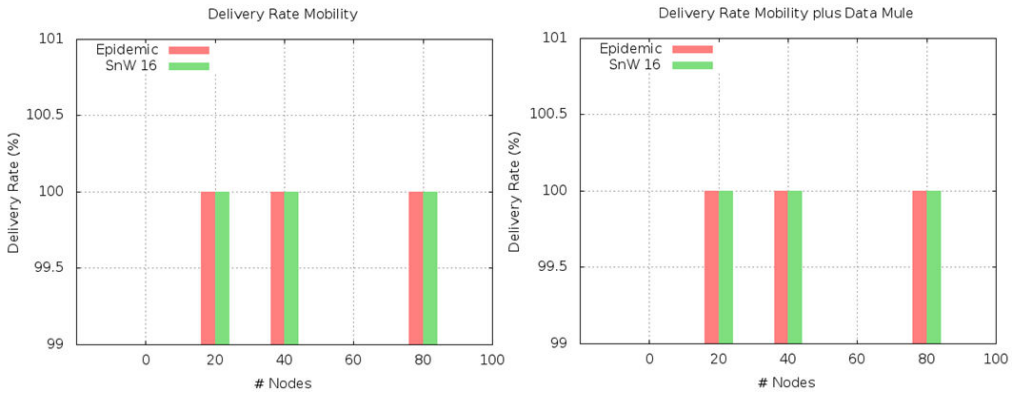

Fig. 5.8 – Retardo de recepción del mensaje en el entorno móvil con 40 nodos.

Para demostrar que la tasa de entrega no siempre es del 100% se han realizado simulaciones complementarias que permiten evaluar en detalle este parámetro. En estas simulaciones se ha utilizado un valor bajo y constante de nodos (veinte) aumentando el área del escenario progresivamente (ver figuras 4.4, 4.8 y 4.9). De esta manera se pretende reducir la densidad de nodos y forzar que algunos paquetes no puedan alcanzar su destino.

De igual manera que para las otras simulaciones se muestran los datos comparativos para el escenario con *Data Mule* intentando mejorar las perdidas obtenidas en el escenario mayor. A continuación se muestra el histograma obtenido.

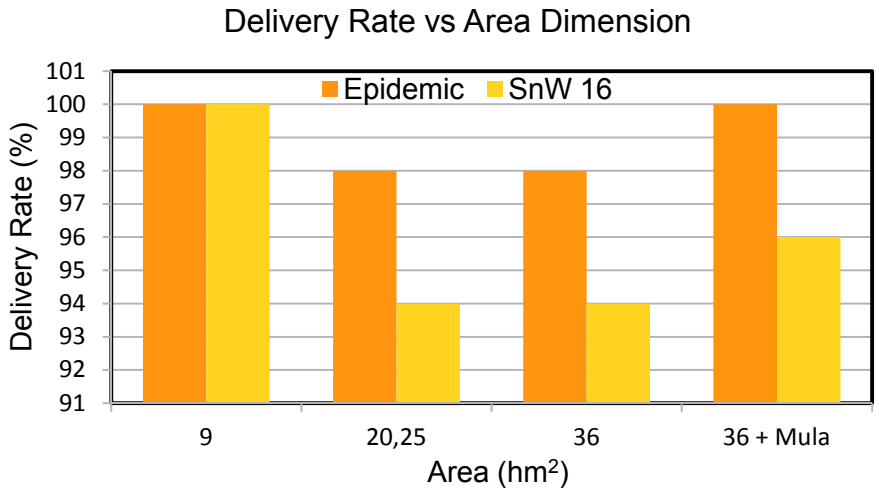

Fig. 5.9 – Retardo de recepción del mensaje en función del área de simulación.

5.2.4 Number of Messages in Environment

En los resultados obtenidos para la ocupación del espacio de almacenamiento total del escenario móvil, igual que para los parámetros anteriores, se comparan con el uso de un nodo tipo *Data Mule*.

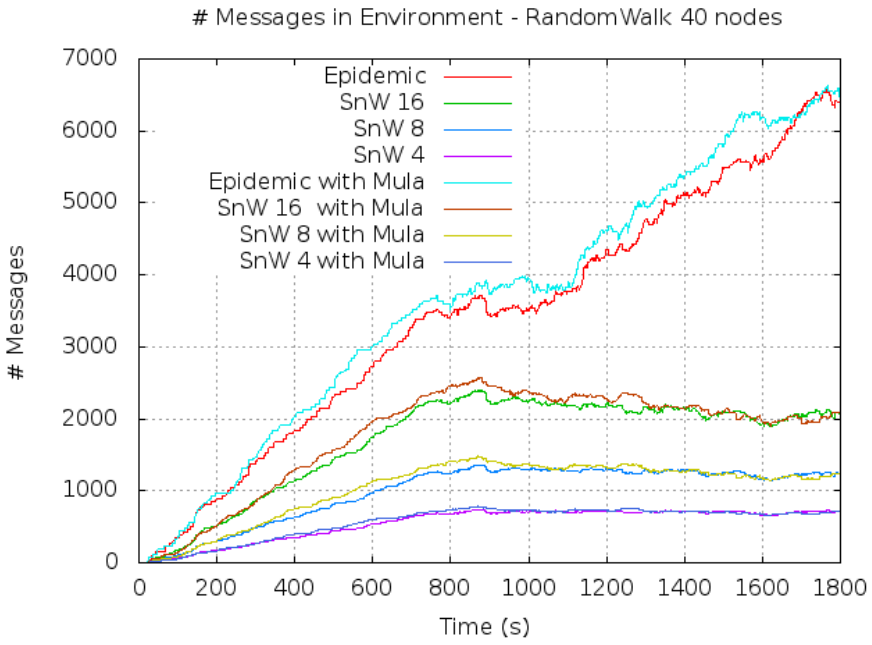

Fig. 5.10 – Evolución de la ocupación de memoria en el entorno móvil.

En la tabla siguiente se muestran los datos obtenidos al final de los 1800 segundos de simulación para cada uno de los entornos propuestos. Las gráficas complementarias se muestran en el apartado B.III de los anexos.

Tabla 5.4 – Datos para el número de mensajes en escenario móvil.

Algoritmo de propagación	**Escenario móvil sin uso de ACK**	
	Sin *Data Mule*	**Con *Data Mule***
Epidemic	6392	6514
SnW16	2077	2092
SnW8	1236	1238
SnW4	709	716

Como resultados complementarios y debido al importante efecto de los *antipackets* en la ocupación de memoria se han realizado simulaciones complementarias para observar su efecto. En el código original se usan estos paquetes para vaciar los buffers de los nodos intermedios una vez el *bundle* ha alcanzado su destino. Al tener este tipo de paquetes deshabilitados en las simulaciones anteriores esta descarga de memoria nunca surgía efecto.

La liberación de memoria se implementa con un paquete tipo *antipacket* en sentido inverso (destino-origen) mediante propagación *Epidemic*. Como se puede comprobar en la siguiente gráfica; con el vaciado de los buffers se consigue disminuir considerablemente la ocupación de menoría en el entorno; llegando a ser cero en algunos instantes de tiempo.

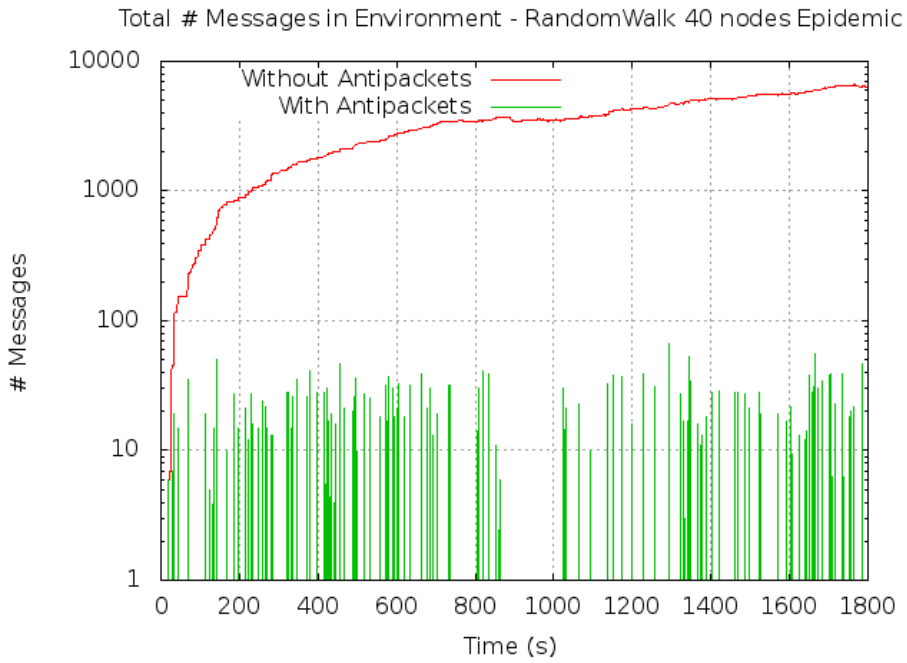

Fig. 5.11 – Evolución de la ocupación de memoria en el entorno móvil con *antipackets*.

CAPÍTULO 6. DISCUSIÓN DE LOS RESULTADOS

6.1 Escenarios estáticos

Las simulaciones con escenarios estáticos son más predecibles. Por lo tanto estas simulaciones han servido para evaluar los cambios implementados y ver su efecto en los parámetros de interés (ver punto 4.1.2). Consideramos por tanto los resultados obtenidos como exploratorios para evaluar y parametrizar el simulador así como los cambios realizados.

La implementación de confirmación de la recepción del mensaje (ACK) no aporta información específica al estudio realizado. Los resultados obtenidos con y sin ACK son extrapolables. El uso de confirmación tipo ACK no tiene ningún impacto en parámetros *Delivery Delay, Delivery Ratio* ni en la ocupación de memoria. Las simulaciones con ACK consumen más energía y requieren una mayor capacidad de proceso del simulador. En general incrementan el tiempo de simulación y no aportan información diferencial de cara al estudio. Por lo tanto, de cara a las simulaciones móviles se decide por defecto realizarlas sin esta confirmación.

Los algoritmos de propagación implementados (*Epidemic y Binary Spray and Wait*) funcionan correctamente. La limitación de los mensajes propagados en el caso *Binary Spray and Wait* funciona según lo esperado. Para ver las diferencias entre algoritmos es suficiente con comparar *Epidemic* con *Spray and Wait* 16. Con un volumen de 40 nodos en los escenarios propuestos se obtiene un coste de simulación contenido y aporta información extrapolable para escenarios mayores. Para ver la evolución del *Delivery Ratio* en función de la densidad de nodos se han utilizado complementariamente simulaciones móviles con 20 y 80 nodos (ver apartado 4.2.2).

6.1.1 Available Energy

Como es de esperar *Epidemic* es el algoritmo que más energía consume para todos los escenarios estáticos propuestos. En el caso de *Spray and Wait*, cuanto menor es el índice de propagación menor es la energía consumida ya que menos copias del mensaje son distribuidas.

Por ejemplo para el caso de 80 nodos estáticos *Epidemic* consume un 14,5% más de energía que *Spray and Wait* 16 (*SnW16*). Sin embargo la diferencia de dividir entre dos la profundidad de *SnW* supone un ahorra de energía de solamente el 2,07%.

Obviamente utilizar ACK implica un pequeño gasto de energía extra (respecto a no usarlo) que en el peor de los casos (*Epidemic,* donde más mensajes se propagan) es del 1,95%.

6.1.2 Delivery Delay

Como se puede observar en la figura 5.2 (y en los correspondientes anexos), el retardo de entrega no es un parámetro que dependa de utilizar o no confirmación de recepción (ACK). Como era de esperar para los entornos propuestos, se obtienen los mismos resultados en ambas pruebas con y sin ACK (ver figura 5.2).

El retardo en la entrega se puede considerar cómo un parámetro de medida de la calidad del algoritmo, a menor retardo mejor es el algoritmo. Esto puede comprometer otros parámetros de eficiencia como por ejemplo el consumo de energía o la ocupación de memoria, entre otros. La propagación tipo *Epidemic* es quien alcanza su destino con mayor rapidez debido a la elevada tasa de propagación. En los casos de *SnW* 16, 8 y 4 las tasas son muy similares (incluso sorprendentes, en el caso SnW4). Esto es debido a que la fuente destino está a pocos saltos del origen y por lo tanto el porcentaje de entrega de SnW4 se maximiza (también debido al gran volumen de copias de mensajes que se propagan por la red, notar que la figura está representada en forma de histograma). Como se puede observar en la figura 5.2 el algoritmo *Epidemic* siempre alcanza el nodo destino con probabilidad máxima para retardos bajos. Por ejemplo en un 96% de los caso alcanzará el destino con un retado inferior a 6 segundos, tal y como se ve en la tabla 6.1 extraída de la figura 5.2. Cabe destacar que estas gráficas si contemplan los paquetes duplicados pero no los retransmitidos.

Tabla 6.1 – Probabilidad de retardo en la entrega según el algoritmo utilizado.

Algoritmo de propagación	**Retardo (%)**	
	< 3 seg.	**< 6 seg.**
Epidemic	56	96
SnW16	48	48
SnW8	47	47
SnW4	64	64

6.1.3 Delivery Ratio

El parámetro *Delivery Ratio* es siempre constante al 100% de entrega para todas las simulaciones realizadas. Debido al objetivo exploratorio de estas pruebas no es conveniente introducir incertidumbres tales cómo pérdidas de los enlaces o situaciones aleatorias no controladas. Se ha forzado por diseño de los escenarios que el destino siempre sea alcanzable. Como se ha comentado anteriormente se necesitan simulaciones complementarias para comprobar que se pueden dar situaciones en las que, en entornos más hostiles, algunos mensajes *bundle* no alcancen su destino (ver puto 6.3). Esto permite demostrar que estos datos son computados por el simulador de forma fehaciente.

6.1.4 Number of Messages in Environment

En las simulaciones realizadas la funcionalidad de vaciado de memoria no está implementada ya que los paquetes tipo *antipacket* han sido deshabilitados. Se observa que el escenario incrementa la custodia de mensajes a lo largo del tiempo, el único mecanismo de vaciado es el tiempo de vida en custodia que cada paquete tiene asignado intrínsecamente en el código base (750 segundos). Es decir, en las simulaciones estáticas realizadas los *bundles* son eliminados solamente 750 segundos después de iniciar la custodia. Esto tiene un efecto negativo en la ocupación de la memoria como se puede ver en la gráfica 5.4 pero positivo de cara al análisis y comparativa de resultados.

Las pequeñas diferencias que se pueden ver en la tabla 5.2 entre enviar o no ACK son debidas tanto al propio ACK (mínimas) como a la aleatoriedad del envío de los *bundles* distribuidos aleatoriamente en el tiempo de simulación (y por tanto la caducidad de los mensajes).

En el caso de 80 nodos (ver gráfica 5.4) con la funcionalidad de vaciado de buffers deshabilitada se observa claramente como la diferencia entre una propagación tipo *Epidemic* y *SnW* es elevada. Al cabo de los 1800 segundos de simulación el escenario con propagación de *SnW16* contiene 11160 mensajes menos que el *Epidemic*, es decir aproximadamente un 84,3% menos. Los escenarios de 40 y 20 nodos tienen respectivamente una ocupación de un 69,2% y un 51,4% inferior. Se observa claramente la limitación de copias del mensaje (de propagación) que el propio algoritmo *Spray and Wait* impone por su naturaleza de funcionamiento.

En el caso de las diferentes profundidades de propagación de *SnW* (16, 8 y 4) se puede observar claramente en la gráfica 5.4 cómo el propio algoritmo limita intrínsecamente el volumen de mensajes de la red. Las líneas de la figura 5.4 que representan *SnW* nunca superan un valor máximo de ocupación (ocupación contenida) así como la línea de *Epidemic* crece de manera continuada en el tiempo de simulación.

6.2 Escenarios móviles

Dados los resultados exploratorios con los escenarios estáticos propuestos en el apartado 6.1, se decide realizar las simulaciones en un entorno de 40 nodos sin ACK. Además se decide no utilizar *antipackets* ya que pueden desvirtuar los demás parámetros evaluados y a su vez implementan un escenario demasiado sintético (en el mundo real si un paquete es descartado “desaparece” de la red). En la implementación realizada al deshabilitar *antipackets* se desactiva implícitamente la funcionalidad de vaciado de *buffers,* con las implicaciones que esto pueda tener en el tiempo de simulación y el consumo de recursos (ver 6.1.4). A pesar de estos inconvenientes, desactivar *antipackets* nos permite observar de manera más clara las diferencias en el consumo de memoria para los algoritmos utilizados.

Cómo se ha comentado anteriormente, el escenario base propuesto (40 nodos móviles, sin ACK ni *antipackets*) se le añade un nodo tipo *Data Mule* para intentar mejorar la profundidad de propagación de los mensajes, es decir, intentar mejorar la calidad de *SnW* que es el algoritmo óptimo en cuanto a consumo de recursos (energía y memoria). Con este nodo de movilidad elevada se pretende mejorar los parámetros de calidad (*Delivery Delay* y *Delivery Ratio*) manteniendo un consumo contenido de los recursos.

Los algoritmos de propagación escogidos han sido *Epidemic* y *Binary Spray and Wait 16.* A continuación se muestran el análisis de los resultados obtenidos para cada uno de los parámetros de evaluación.

6.2.1 Available Energy

En el caso de un escenario móvil se incrementa el consumo de energía ya que un mayor número de copias del mensaje se propaga por la red debido al oportunismo. Si hacemos uso de un *Data Mule* las oportunidades de comunicación se verán incrementadas y consecuentemente el consumo de energía será mayor.

En la figura 5.5 y la tabla 5.3 se pude observar como en el caso *Epidemic* el incremento del gasto energético en el escenario utilizando un nodo *Data Mule* es aproximadamente un 0,5% superior para las simulaciones realizadas. No es un consumo muy elevado, por ejemplo, comparando con los resultados para escenarios estáticos; introducir confirmaciones de recepción de mensajes supone un mayor consumo (ver 6.1.1). El efecto propagador del nodo *Data Mule* no es muy elevado debido a las dimensiones relativamente reducidas del escenario.

Las diferencias entre *Epidemic* y *Spary and Wait* son parecidas a las obtenidas en escenarios estáticos. *Epidemic* es el algoritmo que más energía consume, un 3,8% más que *SnW.* En caso de usar un *Data Mule* este valor se incrementa en 0,2%.

Habilitar los *antipackets* supone un incremento extra del consumo de energía puesto que se envían más mensajes de control. Como se pude ver en la figura 5.6 éste consumo, para el peor de los casos (*Epidemic*), es de aproximadamente un 1,8% superior a la referencia (mismo escenario sin *antipackets*). El consumo de energía de utilizar *antipackets* es superior al que provoca el nodo *Data Mule* propuesto.

6.2.2 Delivery Delay

En la figura 5.7 se ve como *Epidemic* es claramente mejor que *Spray and Wait* para cualquier densidad del escenario. Los mejores retardos de recepción del mensaje siempre siempre se consiguen con propagación *Epidemic*.

La figura 5.7 muestra como *Spray and Wait* incrementa el retardo de entrega del *bundle* debido a la limitación de copias del mensaje distribuidas por la red. El destino es alcanzado con un mayor retardo debido a las limitaciones de propagación propias del algoritmo.

En el caso de 80 nodos se magnifica el efecto estadístico de los retados grandes ya que el número de vecinos es mayor. Por lo tanto hay un mayor número de copias del mensaje y la fuente destino recibirá más copias del mensaje por múltiples caminos.

Tabla 6.2 – Probabilidad de retardo en la entrega según el algoritmo utilizado para el escenario móvil con 80 nodos.

Algoritmo de propagación	**Retardo (%)**	
	< 6 seg.	**< 15 seg.**
Epidemic	46	50
SnW16	31	56

Según se observa en la tabla 6.2 para retardos inferiores a 15 segundos el algoritmo *SnW16* tiene mejor probabilidad de entrega que *Epidemic*. Esto es debido a la reducida dimensión del escenario, al elevado número de vecinos que tiene cada nodo y a los pocos saltos entre origen y destino. Debido al alto volumen de copias del mensaje que generan algoritmos de *flooding* sin limitaciones (caso de *Epidemic*) hace que muchas copias del mensaje (duplicados) lleguen con retardos elevados y esto desvirtúa el volumen de paquetes (no duplicados) recibidos con retardos más bajos (gráficas representadas en forma de histograma).

6.2.3 Delivery Ratio

En ambos algoritmos todos los mensajes enviados llegan a su destino. En las simulaciones complementarias se puede observar como para densidades de nodos más pequeñas el simulador contabiliza las perdidas.

En el caso de un escenario de 36 hm^2 con 20 nodos móviles se reflejan pérdidas de mensajes. Estas pérdidas son más pronunciadas utilizando el algoritmo *Spray and Wait* debido a la limitación de propagación. En este caso se ha demostrado que el uso de un *Data Mule se* puede mejorar la tasa de entrega a costa de un mayor consumo de recursos (energía y ocupación de memoria). Una buena estrategia de *Data Mule* puede solucionar los problemas de conectividad entre regiones de un mismo escenario debido a que maximiza la propagación del mensaje.

6.2.4 Number of Messages in Environment

La ocupación de memoria en los nodos es muy similar respecto al caso del escenario estático (ver gráfica en anexo B.III). Cabe notar que el número de mensajes enviados en todas las simulaciones es el mismo, 100 mensajes distribuidos aleatoriamente en el tiempo de simulación. En el caso del uso de *Data Mule* se observa un incremento del volumen de mensajes en el escenario, esto es debido a que la propagación se ve aumentada oportunistamente y por consecuencia más nodos custodian los mensajes *bundle*.

Tomando un valor discreto, al final de los 1800 segundos de simulación, en el peor de los casos (*Epidemic*) tenemos 122 mensajes más custodiados en el escenario. Esto supone un incremento puntual del 1,9% respecto al escenario sin *Data Mule*.

Tal y como se puede ver en la figura 5.11 al habilitar los *antipackets* el vaciado de *buffers* entra en juego. Por lo tanto hay instantes de simulación en que el número de mensajes custodiados en el entorno es cero; todos los mensajes pendientes han sido entregados y por tanto eliminados de los nodos intermedios. Se da ésta casuística debido a las pequeñas dimensiones del escenario y la cantidad de vecinos que dispone cada nodo. Todos los *antipackets* de vaciado llegan a propagarse por todo el entorno vaciando por completo todos los mensajes custodiados.

CAPÍTULO 7. CONCLUSIONES, IMPACTO MEDIOAMBIENTAL Y LÍNEAS FUTURAS

Los objetivos iníciales del proyecto se han cumplido en su totalidad. En primer lugar se ha validado el correcto funcionamiento del simulador de redes DTN de Aalto para simular Redes Oportunistas. Las herramientas utilizadas son funcionales, la integración de los escenarios generados con *BonnMotion* funciona a la perfección. Los escenarios con uso de *Data Mula* se han realizado de forma manual, es decir se han generado dos escenarios diferentes (escenario de nodos y escenario de *Data Mule)* y se han unido manualmente. Esta tarea no es compleja pero es un poco laboriosa.

Las simulaciones exploratorias se han realizado satisfactoriamente, se han podido reproducir los resultados obtenidos en [22]. Los algoritmos de propagación implementados (*Epidemic* y *Binary Spray and Wait*) son funcionales. La limitación de copias del mensaje en el caso de *Binary Spray and Wait* es configurable y funciona correctamente. La carga de los escenarios de movimientos tipo NS2 se realiza sin problemas.

Se han realizado las modificaciones previstas. La integración con el modulo de energía de NS-3 funciona perfectamente, el output en el log de salida permite, como se ha visto, graficar la cantidad de energía disponible apreciando su evolución temporal. Los datos tipo *bundle* son enviados por defecto con UDP y posen una confirmación a nivel de código (ACK). Desactivar y activar esta confirmación puede ayudar a tener unas simulaciones más livianas (dependiendo del hardware).

El simulador dispone de cómputo de paquetes tipo *antipackets.* Este tipo de paquetes se utilizan para implementar la funcionalidad de vaciado de buffers intermedios. Cuando el *bundle* alcanza su destino se envía un paquete tipo *antipacket* desde el destino hacia la fuente con propagación *Epidemic*. Cómo ya se ha explicado anteriormente uno de los cambios realizados en el código fuente del simulador ha sido desactivar esta funcionalidad. Tal y como se puede observar en el punto 5.2.4 del capítulo anterior, el consumo de memoria se ve claramente mejorado con esta funcionalidad activada pero se dificulta el análisis.

En todas las simulaciones estáticas realizadas la propagación *Spray and Wait* obtiene iguales o mejores resultados que la propagación *Epidemic* para los parámetros de consumo de recursos (energía y memoria) y *Delivery Ratio*. Se puede concluir que *Epidemic* es el algoritmo más dispendioso. En el caso del *Delivery Delay*, *Epidemic* es la estrategia óptima, en iguales condiciones siempre asegura que el destino será alcanzado con un retardo inferior a *Spray and Wait*.

Del estudio realizado en este trabajo se abren varias líneas de investigación. El simulador DTN ofrece la posibilidad de utilizar Control de Flujo (deshabilitado

por defecto). Sería interesante analizar qué impacto tiene esta funcionalidad en los en los parámetros de evaluación con diferentes algoritmos de propagación. El control de flujo asegura que solo se envía un mensaje si el destino dispone de espacio suficiente para su custodia.

Otra línea derivada de este trabajo puede ser implementar más algoritmos de propagación para posteriormente someterlos a evaluación. Algoritmos como PRoPHET, MaxProp o RAPID podrían ser comparados con los resultados obtenidos en este proyecto.

Los escenario móviles escogidos son pequeños y por tanto los nodos tienen un volumen elevado de vecinos. Probar el efecto de los algoritmos en escenarios con una densidad de vecinos inferior podría ser interesante y revelar resultados complementarios. Por ejemplo un escenario con regiones o grupos de movilidad utilizando la funcionalidad de vaciado de ocupación de memoria (*antipackets*) y una estrategia de *Data Mule* oportunista que permita conectar las diferentes regiones. Estos escenarios permitirían evaluar el comportamiento de los algoritmos en redes más realistas.

La realización de este proyecto no tiene ningún impacto directo en la salud de las personas, ni en los recursos naturales. Respecto al medioambiente si se podría tener un efecto positivo. Éste tipo de redes (con un consumo mínimo de energía y recursos) se pueden desplegar en parajes naturales o lugares de difícil acceso con la finalidad de monitorizar el medioambiente. Estas prácticas ya se están realizando con redes de sensores ad-hoc donde normalmente los nodos suelen ser estáticos; añadir movilidad a estas redes pude multiplicar sus aplicaciones y resultados obtenidos (por ejemplo en la fauna).

Otro de los efectos indirectos en la salud y el desarrollo humano se pueden ver claramente identificados en [12], donde se estudia como dar conectividad de datos a poblaciones nómadas aisladas mediante el oportunismo.

BIBLIOGRAFÍA

[1] Chung-Ming Huang, Kun-chan Lan and Chang-Zhou Tsai “A Survey of Opportunistic Networks”, 22nd International Conference on Advanced Information Networking and Applications – Workshops (2008).

[2] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass and James Scott “Impact of Human Mobility on Opportunistic Forwarding Algorithms”, Mobile Computing, IEEE Transactions (June 2007).

[3] Yu-Chee Tseng, Fang-Jing Wu and Wan-Ting Lai, “*Opportunistic data collection for disconnected wireless sensor networks by mobile mules*”. Sience Direct (May 2013).

[4] Luciana Pelusi, Andrea Passarella, and Marco Conti “Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks”. Communications Magazine, IEEE. Volume 44, Issue 11. (November 2006).

[5] Missouri University of Sience and Technology, Mobile Data Management and Application “Delay Tolerant Networks”. http://www.mst.edu/ (May 2006).

[6] Yue Cao and Zhili Sun, “*Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges*”. IEEE Communications Surveys & Tutorials, Vol. 15, No. 2 (Second Quarter 2013).

[7] T. Small and Z. J. Haas, “The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way),” in ACM MobiHoc’03, Annapolis, Maryland, USA (2003).

[8] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, “Pocket switched networks and human mobility in conference environments,” in ACM WDTN’05, Philadelphia, Pennsylvania, USA (2005).

[9] P. Pereira, A. Casaca, J. Rodrigues, V. Soares, J. Triay, and C. Cervello-Pastor, “From delay-tolerant networks to vehicular delay-tolerant networks,”IEEE Commun. Surveys Tuts. , vol. PP, no. 99, pp. 1 –17 (2011).

[10] T. Jonson, J. Pezeshki, V. Chao, K. Smith, and J. Fazio, “Application of delay tolerant networking (dtn) in airborne networks,” in IEEE MILCOM ’08, San Diego, California, USA (2008).

[11] A. Pentland, R. Fletcher, and A. Hasson, “Daknet: rethinking connectivity in developing nations,” IEEE Computer, vol. 37, no. 1, pp. 78 – 83 (2004).

[12] Avri Doria, Maria Uden and D urga Prasad Pandey. “Providing connectivity to the Saami nomadic community”. 2nd Int. Conf. on Open Collaborative Design for Sustainable Innovation (2002).

[13] H. Nguyen and S. Giordano, "*Routing in opportunistic networks*". International Journal of Ambient Computing and Intelligence (IJACI) (2009).

[14] C.-M. Huang, K.-c. Lan, and C.-Z. Tsai, "A survey of opportunistic networks" In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications – Workshops (2008).

[15] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani, "DTN routing as a resource allocation problem" In Proc. ACM SIGCOMM (August 2007).

[16] Thrasyvoulos Spyropoulos, Konstantinos Psounis and Cauligi S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks" SIGCOMM'05 (2005).

[17] Página oficial de Nerwork Simulator 3. http://www.nsnam.org.

[18] Página oficial de la distribución de Linux Ubuntu http://www.ubuntu.com

[19] Página oficial de la universidad finlandesa Aalto en Helsinki fundada en 2010 http://www.aalto.fi

[20] http://www.netlab.tkk.fi/tutkimus/dtn/ns/

[21] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC Experimental 5050, IETF (November 2007).

[22] Jani Lakkakorpi and Philip Ginzboorg, "ns-3 Module for Routing and Congestion Control Studies in Mobile Opportunistic DTNs". Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2013 International Symposium in Toronto (July 2013).

[23] Modelo de Energia NS-3.
http://www.nsnam.org/wiki/index.php/Energy_model

[24] Página oficial de BonnMotion (herramienta de generación de escenarios), http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANEXOS

TÍTULO DEL PFC: Evaluación de algoritmos de propagación de mensajes en redes oportunistas

TITULACIÓN: Ingeniería de Telecomunicaciones (segundo ciclo)

AUTOR: David García Robles

DIRECTOR: Roc Messeguer Pallarès

FECHA: 3 de Noviembre de 2013

A. Resultados Complementarios Escenarios Estáticos

A.I Available Energy in Environment

A continuación se muestran la evolución de la energía disponible en el entorno para las simulaciones estáticas exploratorias con 20 y 40 nodos.

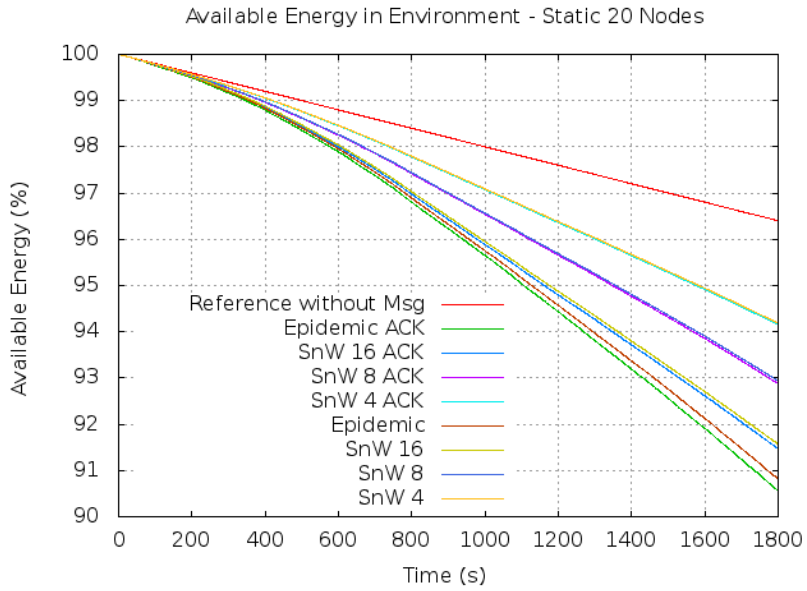

Fig. A.1 – Evolución de la energía disponible en el entorno para 20 nodos.

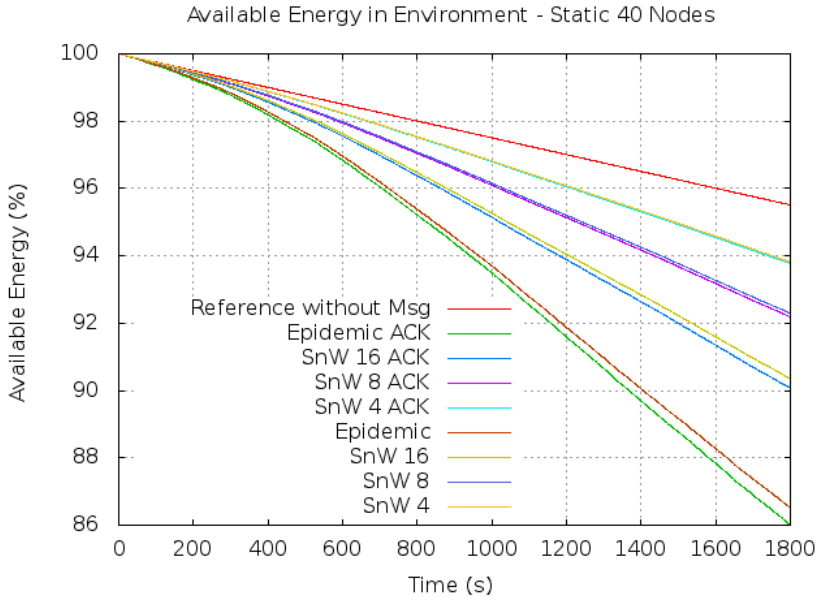

Fig. A.2 – Evolución de la energía disponible en el entorno para 40 nodos.

A.II Delivery Delay

En las figuras A.3 y A.4 se muestran los histogramas de retardo en la entrega para los escenarios con 20 y 40 nodos.

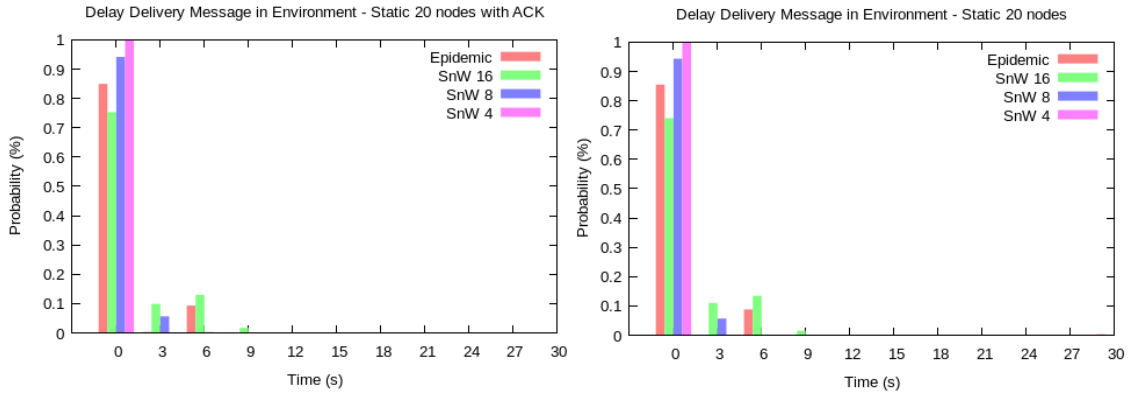

Fig. A.3 – Retardo de recepción del mensaje para escenario estático 20 nodos.

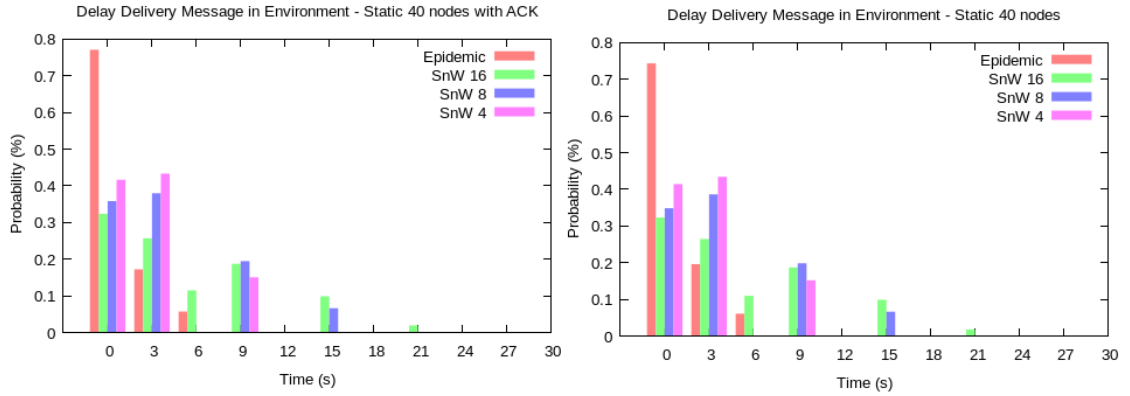

Fig. A.4 – Retardo de recepción del mensaje para escenario estático 40 nodos.

A.III Number of Message in Environment

En las figuras A.5 y A.6 se muestra la evolución temporal de la ocupación de memoria en el entorno para los escenarios estáticos de 20 y 40 nodos respectivamente (con uso de ACK). Las figuras A.7 y A.8 muestran estos mismos resultados sin uso de confirmación del *bundle* tipo ACK.

Cabe notar que estos resultados han sido obtenidos con la funcionalidad de vaciado de memoria desactivada, tal y como se ha explicado en el punto 3.2.4 de este documento.

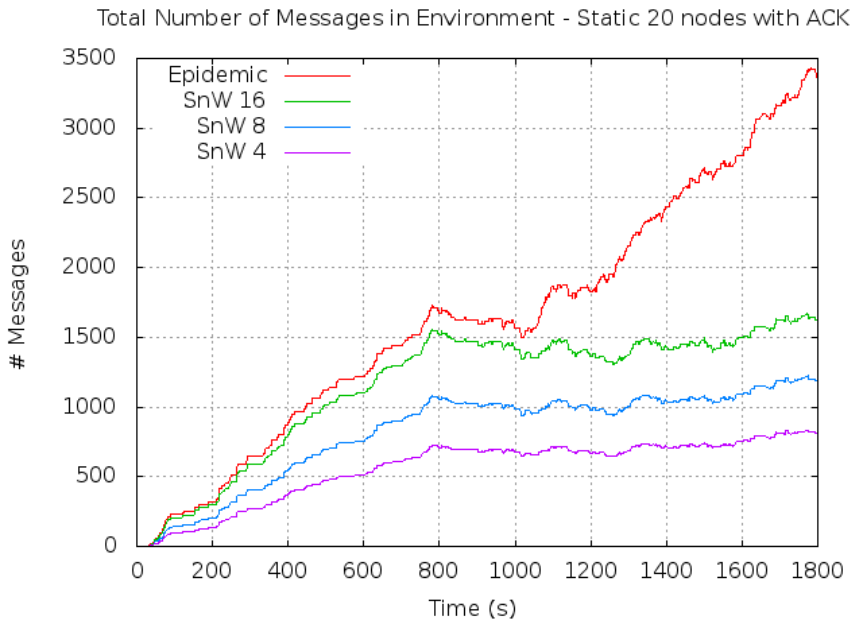

Fig. A.5 – Evolución de la ocupación de memoria en el entorno para escenario estático con 20 nodos con ACK.

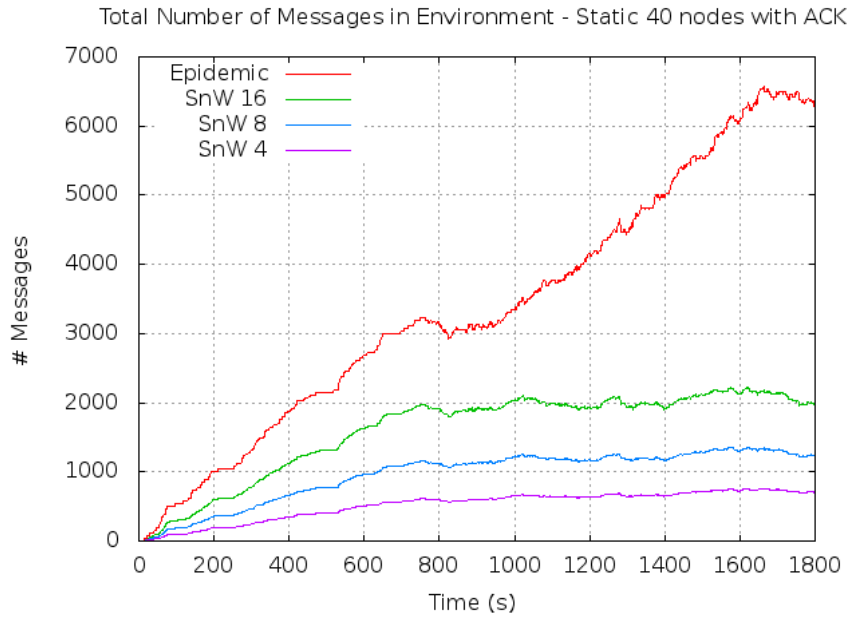

Fig. A.6 – Evolución de la ocupación de memoria en el entorno para escenario estático con 40 nodos con ACK.

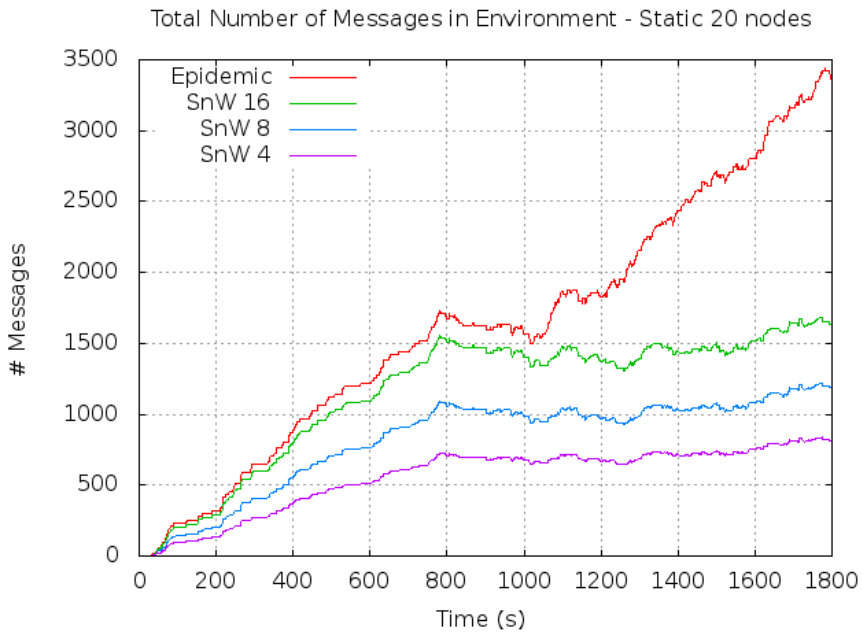

Fig. A.7 – Evolución de la ocupación de memoria en el entorno para escenario estático con 20 nodos.

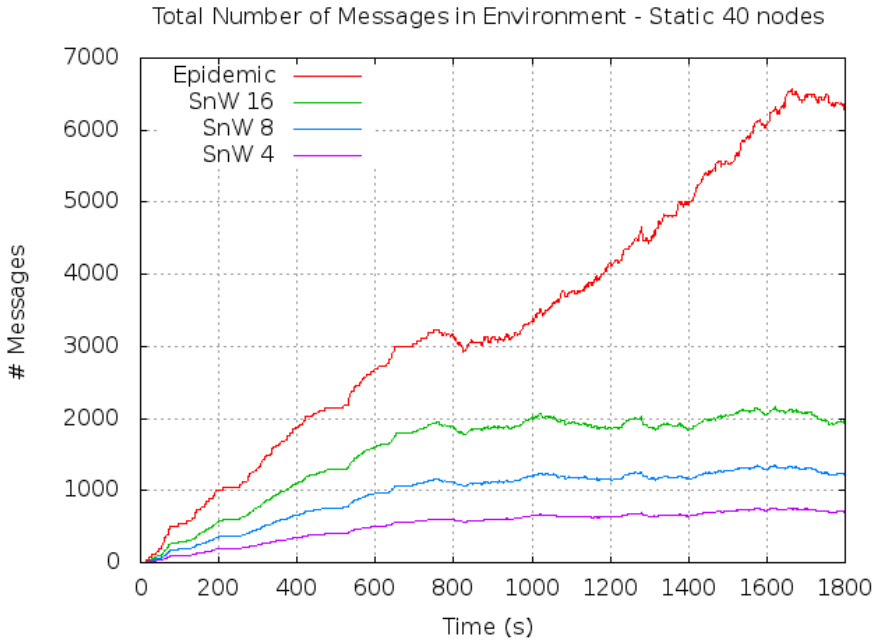

Fig. A.8 – Evolución de la ocupación de memoria en el entorno para escenario estático con 20 nodos.

B. Resultados escenarios con movilidad

Se han realizado algunas simulaciones complementarias al escenario de 40 nodos propuesto. A continuación se muestran los resultados obtenidos.

B.I Available Energy in Environment

En las figuras B.1 y B.2 muestran los resultados obtenidos para los escenarios móviles con 20 y 80 nodos respectivamente.

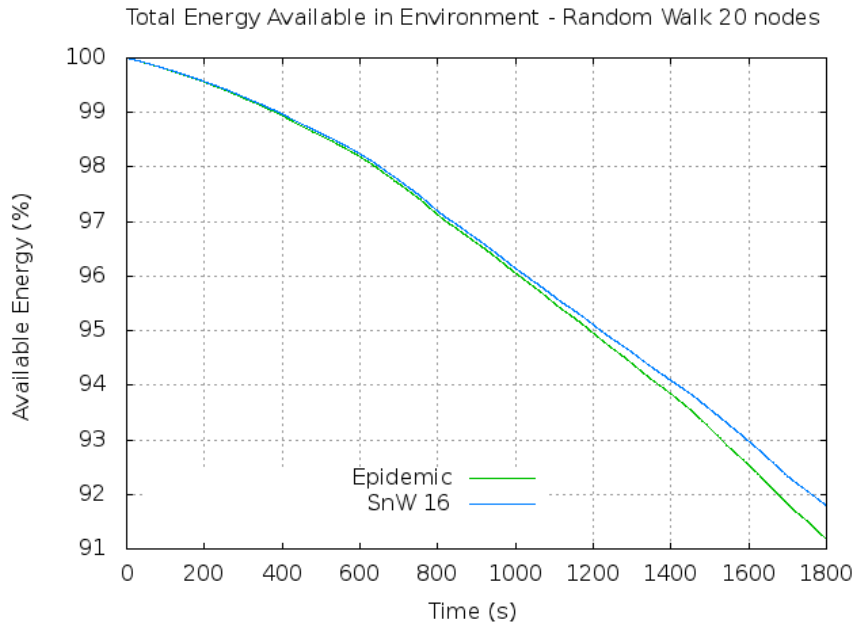

Fig. B.1 – Evolución de la energía disponible en el entorno móvil con 20 nodos.

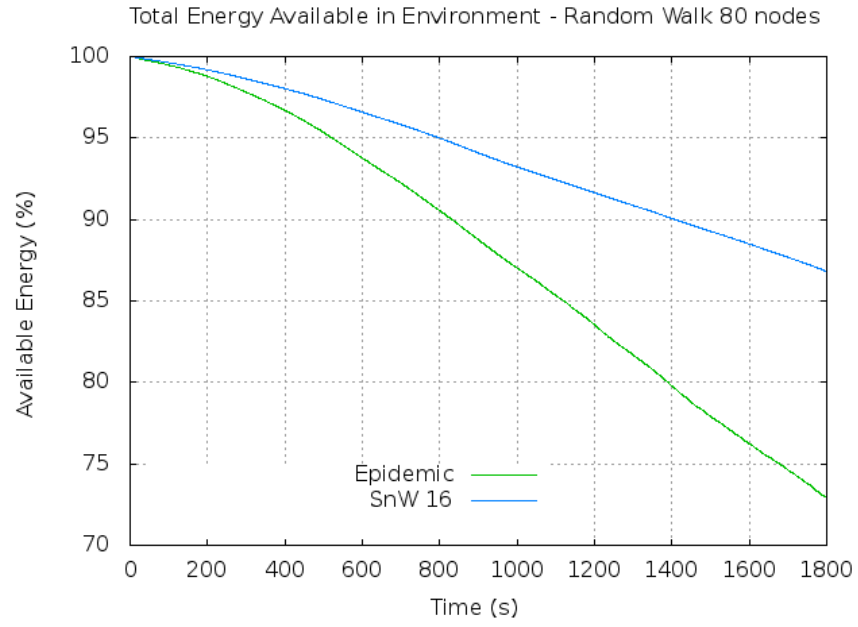

Fig. B.2 – Evolución de la energía disponible en el entorno móvil con 80 nodos.

B.II Delivery Delay

La figura B.3 muestra el histograma de retardos en el escenario móvil de 40 nodos propuesto para todos los algoritmos de propagación probados.

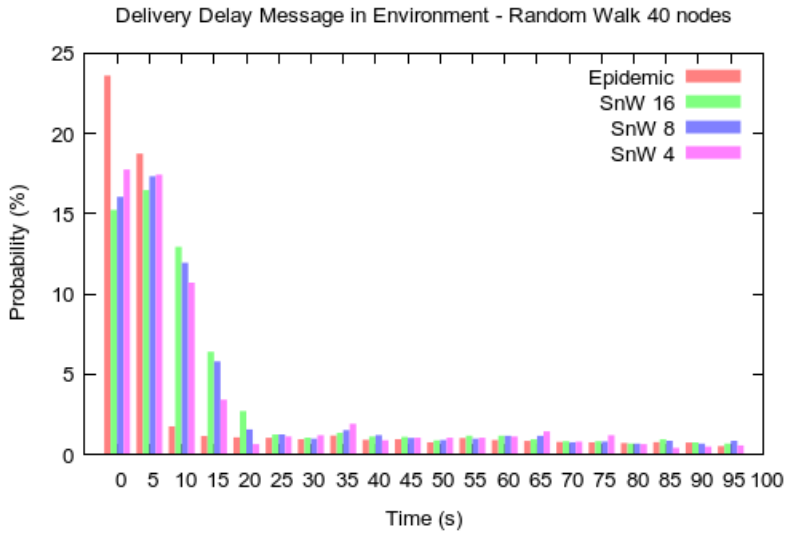

Fig. B.3 – Histograma del retardo de recepción del mensaje en entorno móvil.

En las figuras B.4 y B.5 presentan los histogramas para los escenarios móviles con 20 y 80 nodos respectivamente. En estos escenarios no se ha probado de añadir un nodo *Data Mule*. Como se ha comentado anteriormente los resultados para 40 nodos pueden ser extrapolables a escenarios con diferentes densidades.

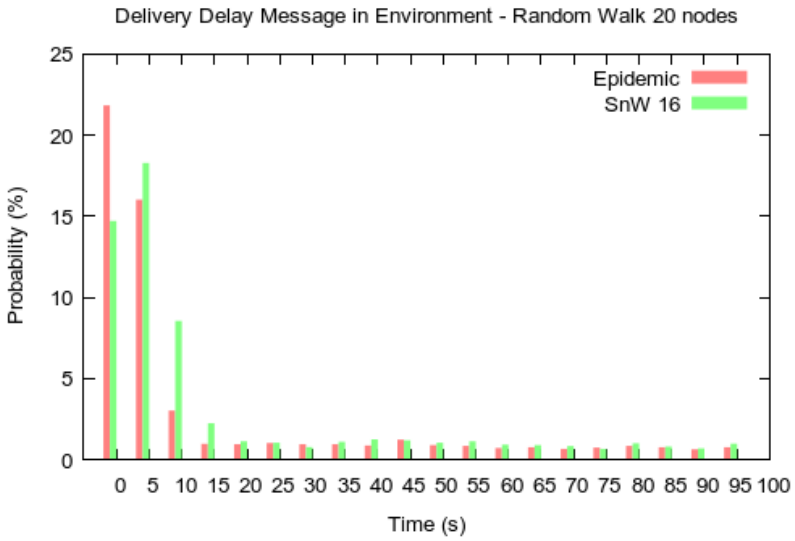

Fig. B.4 – Histograma del retardo de recepción del mensaje en entorno móvil con 20 nodos.

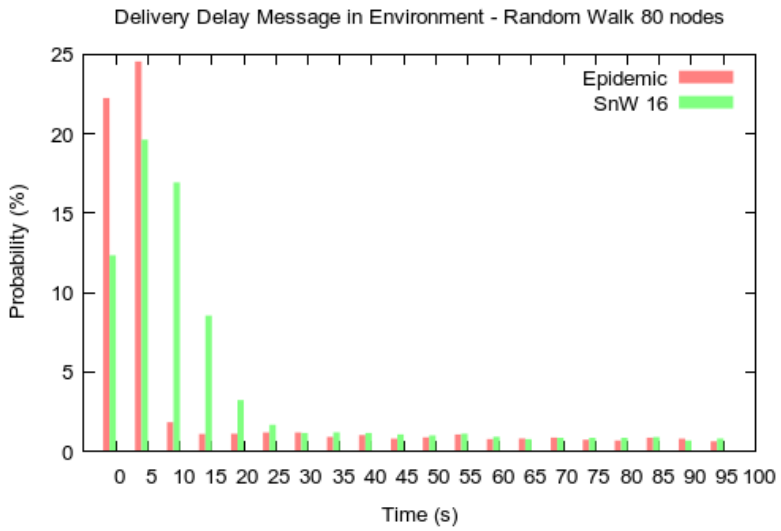

Fig. B.5 – Histograma del retardo de recepción del mensaje en entorno móvil con 80 nodos.

B.III Number of Messages in Environment

Las figuras B.6 y B.7 muestran la evolución temporal de la ocupación de memoria en el entorno para los escenarios móviles con 80 y 20 nodos respectivamente (sin uso de ACK).

Cabe notar que estos resultados han sido obtenidos con la funcionalidad de vaciado de memoria desactivada, tal y como se ha explicado en el punto 3.2.4 de este documento.

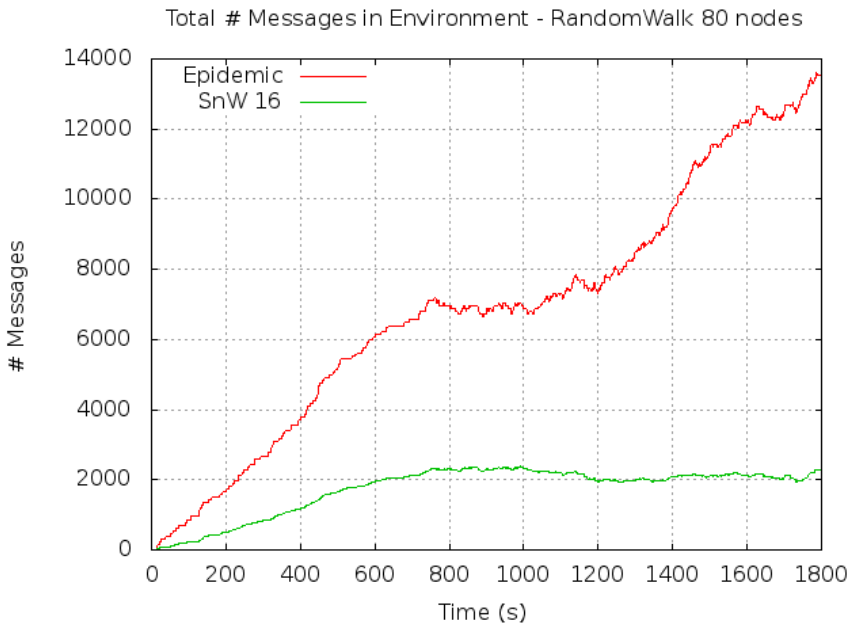

Fig. B.6 – Ocupación de memoria en entorno móvil con 80 nodos.

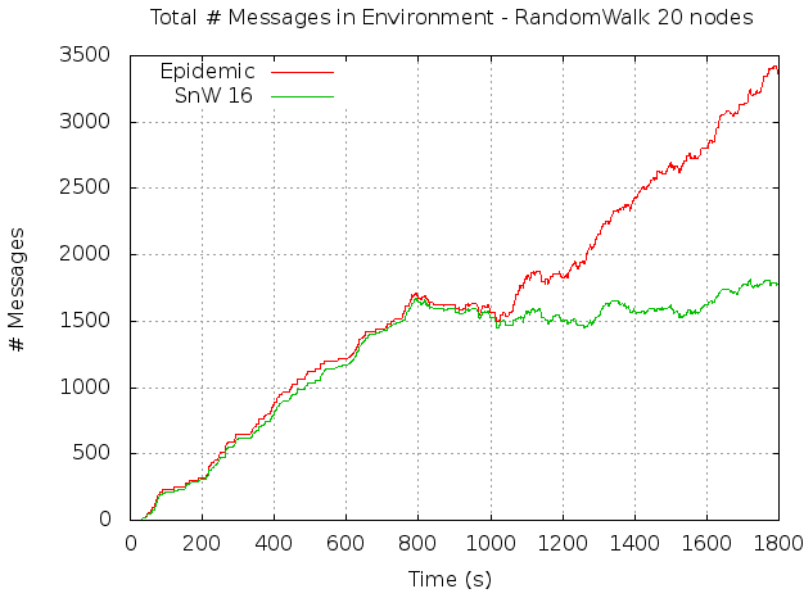

Fig. B.7 – Ocupación de memoria en entorno móvil con 20 nodos.